C
Badger

reference manual MODCOMP II computer

modular computer systems . . . our name, our claim.

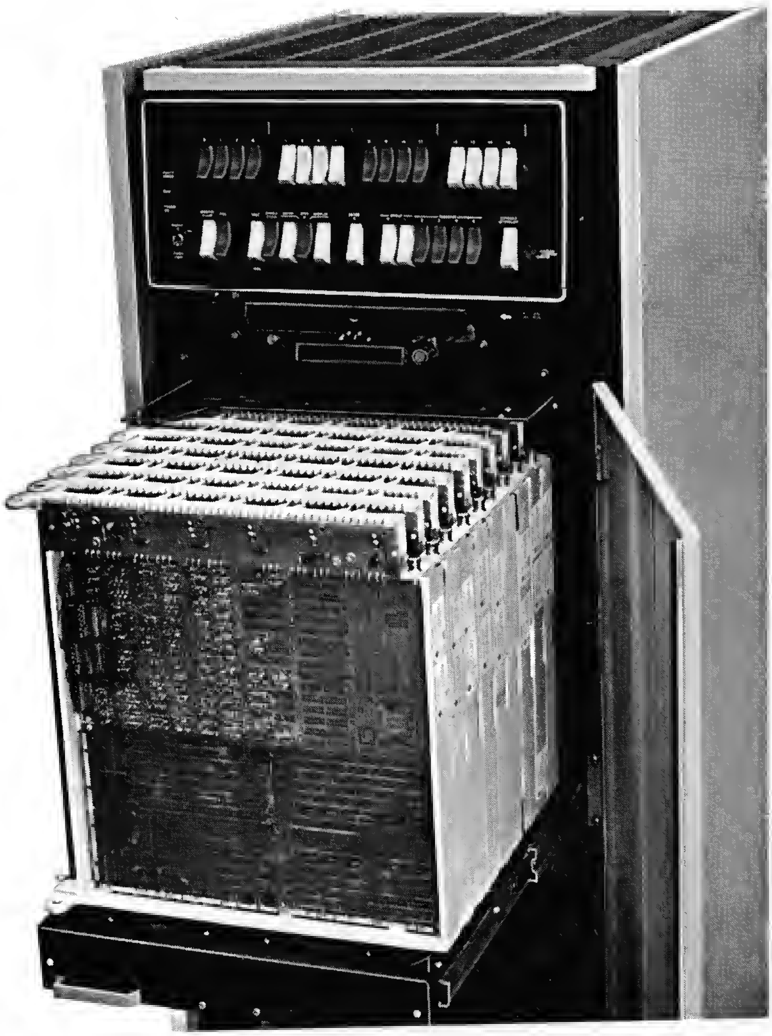
MODCOMP II COMPUTER

REFERENCE MANUAL

OCTOBER 1972

210-102000-000
PRICE: $7.00

Modular Computer Systems
1650 West McNab Road
Fort Lauderdale, Florida 33309
(305) 974-1380



MODCOMP II Computer with Rack open and Planes exposed.

CONTENTS

FIGURES

TABLES

APPENDICES

I. MODCOMP II CHARACTERISTICS

MODCOMP II is an 800-nanosecond, 16-bit computer having many of the characteristics of 32-bit computers. It is designed to be efficient in executing higher level software including real-time multiprogramming systems, disc operating systems and interactive language systems. The large instruction set and general register file also permit highly efficient machine code to be written.

MODCOMP II consists of a set of functional modules implemented with the present state of the art in IC, MSI, LSI and core memory technology designed to be upgraded when new advances in component technology are available. All data transfers and manipulations within the computer are controlled by a highly-flexible solid state LSI memory (ROM) controller. The ROM controller provides a rich instruction set including bit, byte, word, doubleword, tripleword (including floating point) and file manipulation instructions.

The MODCOMP II is the intermediate member of the MODCOMP computer family. It has the same standard instruction set as the larger MODCOMP III computer. The same set of optional instructions is also available with both computers. Therefore, all of the software available with MODCOMP III is also available with MODCOMP II.

MODCOMP II is a superset of MODCOMP I. All MODCOMP I programs are executable in MODCOMP II.

GENERAL CHARACTERISTICS

The organization of MODCOMP II is shown in Figure 1-1.

The machine is packaged in two basic versions. The MODCOMP II/5 contains two CPU planes and either one or two memory planes. Each memory plane can contain a 4K, 8K, or 16K word core memory module or 512 to 2,048 words of solid state memory.

The MODCOMP II/20 and higher numbered models contain two CPU planes, one to four memory planes, and up to three option planes. The block diagram designates which options are available only in the larger computer package.

MODCOMP II consists of storage, processing and input/output modules and a modular bus through which all inter-module transfers are made. The major features of each module are described on the following pages.

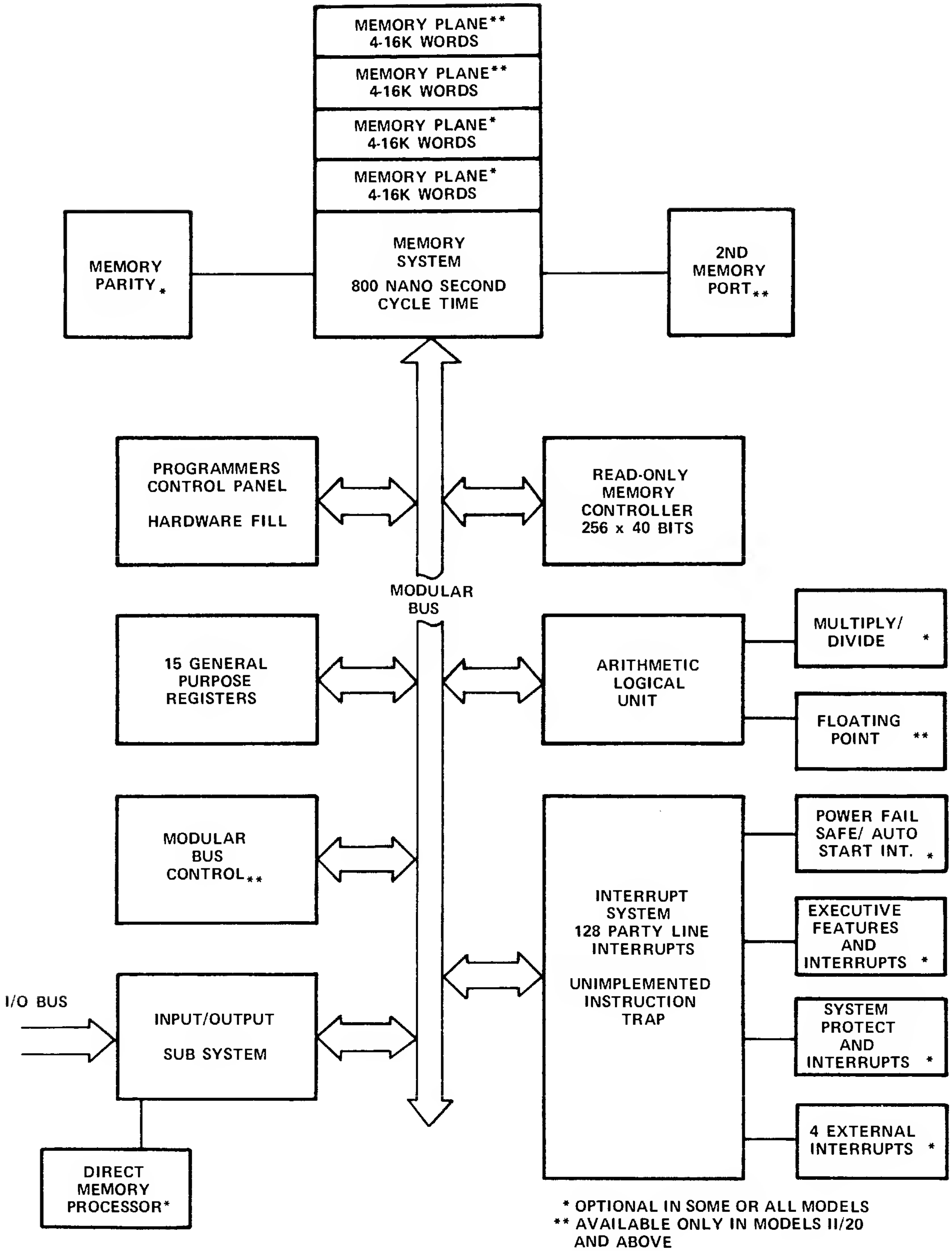


Figure I-I MODCOMP II BLOCK DIAGRAM

Memory System

- 4,096 to 65,536 16-bit words, expandable by 4K, 8K, or 16K word modules
- 400 nanosecond access time
- 800 nanosecond full cycle time
- All memory locations directly addressable
- Seven memory addressing modes provided including indirect, indexed and immediate
- Dual, concurrent access available in multiprocessor configurations
- Memory Protect option
- Memory Parity option

General Register File

- 15 addressable, 16 bit, general purpose registers
- 7 of the general registers usable as index registers
- All 15 registers usable for short indexing operations
- 800 nanoseconds execution time for typical register-to-register instructions

Arithmetic Module

- Parallel operation
- Full set of arithmetic, logical, compare, and shift capabilities
- Execution times

  Add, Subtract, And, Or, Exclusive Or (Reg.-to-Reg.) = 0.8 μ sec.
  Add, Subtract, And, Or, Exclusive Or (Mem.-to-Reg.) = 1.6 μ sec.
  Multiply (Reg.-to-Reg.) = 6.7 μ sec., (Mem.-to-Reg.) = 7.2 μ sec.
  Divide (Reg. by Reg.) = 11.0 μ sec., (Reg. by Mem.) = 11.4 μ sec.
- Implemented with four MSI modules

Read-Only Memory Controller

- 267 nanosecond cycle time
- 40-bit word length
- 256 words in basic computer
- Optional instructions including floating point arithmetic and fixed point multiply/divide

Input/Output System

- Program controlled transfers to/from 63 peripheral devices
- Transfers synchronized by interrupts
- Transfers can be made from any general register to any device
- Transfers are made over a buffered input/output bus which isolates the computer from external cable and controller delays
- Direct Memory Processor available for automatic block transfers to/from 8 peripheral devices on a multiplexed basis
- Controller for ASR-33, ASR-35, or KSR-35 Teletype
- High-Speed paper tape reader, in addition to Teletype, can be operated from the controller.

Modular Bus Control

- Permits Direct Memory Access transfers at rates up to 1.25 M words per second
- Makes all machine processing logic available to an external controller
- Enables custom macro instructions to be added to the CPU using ROM control

Interrupt System

- Up to 16 unique priority levels
- Two standard input/output levels
- Each of these two priority levels (C,D) are connected to 17 unique sub-priority levels which can be connected to up to 64 sublevels, each with a unique (dedicated) memory pointer.
- Standard unimplemented instruction trap
- Complete program control of the Request, Enable*, and Active states of each level
- System Protect Feature includes memory protect and privileged instruction trap capabilities which enable the computer to operate in either a protected or an unprotected mode.
- Executive features includes a real-time clock (200 Hz), console interrupt, task scheduler interrupt, and floating point overflow interrupt

Control Panel

- Capability to display or modify the contents of any memory location, general register or most non-programmable registers
- Program fill switch
- Control panel lock switch
- Master Clear to clear computer and peripherals
- Optional Console Interrupt

Physical Characteristics

- 0-55°C operating ambient temperature range
- 120 $\pm$ 10% VAC, 48 to 62 Hz
- Packaged for mounting in a standard 19-inch cabinet. Occupies 8.75" (II/5) or 21" (II/20) vertically.

MODCOMP SOFTWARE

Executive Systems

Three Modular Application Executive (MAX) systems are available with MODCOMP II computers to meet the requirements of a wide range of machine operating environments. A Special Application Executive (SAX) is also available for dedicated real-time applications.

MAX I is a core resident operating system which improves machine utilization efficiency in assembling, debugging and related operations.

*Internal error exception interrupts are always enabled.

MAX II is a disc operating system which accepts a batch job input consisting of assemblies, compilations and/or executions. A core resident version is also available for non-disc systems.

MAX III is a real-time multiprogramming executive which provides complete task scheduling, initiation, termination and I/O services. This system will control the execution of any mixture of foreground/middleground and background tasks. Unprotected (middleground) tasks can be brought on-line without disturbing other protected (foreground) tasks. Batch processing can be performed in the background. A core-only version is available for dedicated applications.

SAX is a real-time executive which provides task scheduling, I/O services and a flexible operator communications package.

Language Processors

Several language processors are available with MODCOMP systems.

FORTRAN IV - The MODCOMP FORTRAN compiler meets the ANSI FORTRAN specifications. It is designed to produce efficient code by using all machine capabilities such as all registers in the register file and all instructions. It produces assembly language output, permitting the programmer to optimize further. The programmer can also write programs in any desired mixture of compiler and assembly languages. It is available in a core-resident or overlay version under MAX II and MAX III.

Extended Fortran IV - This FORTRAN compiler is an extension of FORTRAN IV as defined above containing random access I/O operations through DEFINE FILE. This compiler contains block level optimization to produce efficient object code. It is available in core-resident or overlay versions under MAX II or MAX III.

BASIC - This multi-user system is a subset of the Dartmouth BASIC system operating under either MAX II or MAX III. It enables users having no previous programming experience to write programs in a simple, quickly learned language.

Macro Assembler - This big machine class assembler has an extensive set of directives and error diagnostics as well as a macro processor. It accepts conditional assembly statements, assembly time branches and macro exits. It is a two-pass assembler, operating under MAX II and MAX III. It is available in core-resident or overlay versions.

Assembler - The assembler is a subset of the macro assembler. It generates relocatable as well as absolute object code and operates under MAX I, II or III.

FORTRAN Coded Assembler - The assembler is available in FORTRAN source language. This assembler operates in IBM 360/370 or CDC 6000 series computers and is compatible with the MODCOMP II ASSEMBLER in both syntax and binary output. The user can therefore assemble programs on larger machines then run them on the MODCOMP II with no modifications. It operates under OS or DOS in 65K bytes.

Compatible Assembler - This assembler accepts MODCOMP I Assembler source code and produces object code executable in a MODCOMP I, II or III computer.

Diagnostics, Utilities, Math Library

An advanced set of computer and peripheral diagnostics are available as maintenance aids. Utilities include source and object file editing, media-to-media conversion, and program debug capabilities. The math library meets ANSI FORTRAN standards.

MODCOMP DATA PROCESSING PERIPHERALS

Modular peripherals are available for a broad spectrum of applications including program preparation, data processing and system support functions. All peripherals are supported by the appropriate MAX system. The basic specifications for each device are summarized below.

Page Printers	- ASR-33, ASR-35, KSR-35 Teletypes
Paper Tape Reader	- 625 characters per second
Paper Tape Reader and Punch	- 625 characters per second read, 110 characters per second punch
Card Readers	- 300-1000 cards per minute
Card Punch	- 100 cards per minute
High Speed Serial Printer	- 50-150 lines per minute, 132 columns
Line Printers	- 600 lines per minute, 80-132 columns
Magnetic Tape Units	- 12.5/45 IPS, 7/9 track, 556/800 BPI, industry compatible NRZ. 45 IPS, 9 track, 1600 BPI industry compatible Phase Encoded.
Moving-Head Discs	- 20 millisecond average latency Capacity range - over 1.2M, 13M and 26M words Transfer rates - 97.8K words and 156K words per second

MEASUREMENT, CONTROL AND COMMUNICATION EQUIPMENT

A complete range of analog input, analog output, digital input, digital output and communication equipment is available to operate with MODCOMP computer systems. This equipment has all been designed together expressly to operate with MODCOMP systems. Therefore, hardware formats, interfaces, cabling and power supplies are the same in all units to facilitate customer usage and minimize spares requirements.

High Level Analog Input Subsystem

Channel Capacity	- 16-128 Channels single-ended or 8-128 Channels differential
Input Range	- ±10.24 volts full scale or ±102.4 volts full scale
Throughput Rate	- 50,000 Channels per second max.
Overall Accuracy	- ±0.05% Full Scale ±1/2 LSB

Wide Range Solid State Analog Input Subsystem

Channel Capacity	- 8-128 Channels
Input Range	- 12 Programmable ranges from $\pm$5 MV to $\pm$10.24V Full Scale
Throughput Rate	- 20,000 Channels per second max.
Overall Accuracy	- $\pm$0.05% Full Scale $\pm$1/2 LSB
Auto Ranging	- With 4,000 Channels per second throughput
Zero Suppression	- Optional

Wide Range Relay Analog Input Subsystem

Channel Capacity	- 8-512 Channels
Input Ranges	- 12 Programmable ranges from $\pm$5 MV to $\pm$10.24V Full Scale
Throughput Rate	- 200 Channels per second max.
Overall Accuracy	- $\pm$10 Microvolts or $\pm$0.05% Full Scale
Auto Ranging	- Standard
Zero Suppression	- Optional

Input/Output Interface Subsystem

Channel Capacity	- 16 Input/Output channels of 16 bits each plus expander chassis (up to 2048, 16 bit channels)
Digital Inputs	- Micrologic, positive voltage, negative voltage, bipolar voltage, contact sense. (Isolated and filtered inputs)
Digital Outputs	- Micrologic, positive voltage, negative voltage, electronic switch, contact closure, pulse output, and AC output (TRIAC)
Analog Outputs	- 12 Bits binary, including sign
	- $\pm$10 volts, $\pm$20 volts, 1 to 5 ma, 4 to 20 ma, 10 to 50 ma
Serial Communications Interface	- RS 232 or 20 ma current loop (TTY compatible)
Interval Timer	- Provides programmable timing interrupt or 'watchdog' timer
I/O Interrupts	- Provides 8 data interrupts and/or 8 service interrupts
External Interrupts	- Provides signal conditioning and drivers for 16 external interrupts
Synchronizer	- Provides 'handshake' data transfer

MODAC Subsystem

Provides a flexible combination of analog and digital data acquisition modules.

Analog Input Module	- 32 High level ($\pm$10.24V) channels, 20K SPS
Analog Output Module	- Eight D-to-A converters, 12 bits binary, current and voltage outputs
Dual Word Input Module	- Two 16-bit digital input channels
Dual Word Output Module	- Two 16-bit digital output channels

Communications Multiplexers

Types	- Universal, operates in synchronous and/or asynchronous mode. Asynchronous, operates in asynchronous mode only.
Channel Capacity	- Universal, 4 to 32 full duplex channels expandable in groups of 4 up to 64 full duplex channels. Asynchronous, 2 to 32 full duplex channels expandable in groups of 2 up to 128 full duplex channels.

Communications Channels

ASYNCHRONOUS

Clocking Mode	- Asynchronous
Communication Interfaces	- EIA RS-232-C Modems, TTL Modems, TTY 60/20 ma Current loop
Baud Rate	- Patchable from 75 to 9600 baud with a maximum of five different baud rates per multiplexer
Codes	- Program selectable - 5, 6, 7, or 8 bits plus parity
Stop bits	- Program selectable - 1 or 2
Parity	- Program selectable - none, odd, even
Echo	- Program selectable - Echos on full duplex line

SYNCHRONOUS

Clocking Mode	- Synchronous
Communications Interfaces	- EIA RS-232-C Modems, TTL Modems
Baud Rate	- Patchable to 50K baud with a maximum of five different baud rates per multiplexer
Code	- Program selectable - 5, 6, 7, or 8 bits plus parity
Parity	- Program selectable - none, odd, even
Sync Character	- Patchable

SYSTEM EXPANDABILITY

The modular design makes the MODCOMP II computer easily expandable. The 21 inch high assembly is capable of containing all system features. Core memory up to a total of 64K words is available on a total of four memory planes. The interrupts are also modular and are field expandable. Even the concurrent memory access path (second port) can be added in the field. Therefore, a MODCOMP system can be expanded as system requirements grow. It can even be converted into a multiprocessor system if the need for a substantial increase in computing capability arises.

MULTIPROCESSOR CONFIGURATIONS

The MODCOMP II is available as a multiprocessor having two CPU's and both private and shared memory modules. The range of multiprocessor configurations available is shown in Figure 1-2.

Each of the two computer cabinets can contain from 16K to 64K words of memory, and each CPU can address up to 64K words. Memory can be connected to the CPU in the other cabinet on a 16K word basis, except for the highest memory section which can be 4K, 8K or 16K words.

The lower 16K memory section cannot be shared because the lower memory locations in each computer are dedicated interrupt and I/O locations. Either one shared or two private modules can be connected between the 16K and 32K address boundaries and also between the 32K and 48K boundaries.

The CPU-to-CPU communication interrupt is generated by execution of the Request Multiprocessor Interrupt instruction. Whenever this instruction is executed in one CPU, an interrupt signal is sent to Level 3 in the other CPU.

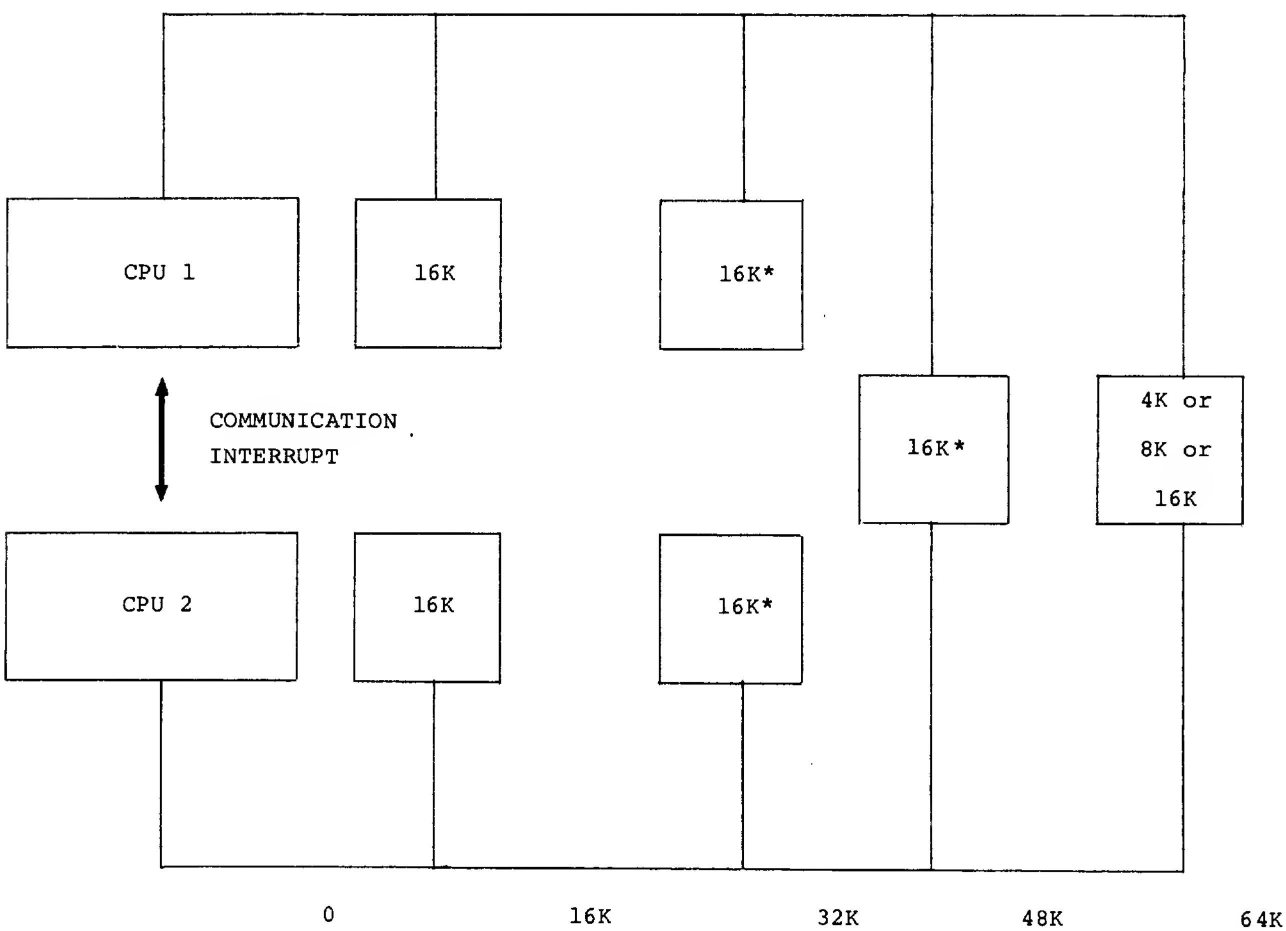


FIGURE 1-2

* Can be private or shared

II. CENTRAL PROCESSOR DESCRIPTION

INFORMATION FORMATS

Basic Formats

The 16-bit word is the basic information format of the MODCOMP II computer. The bit designations in the computer word are:

WORD FORMAT

WORD

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Some instructions operate on doublewords which consist of 32 bits of data stored in two consecutive register or memory locations.

DOUBLEWORD FORMAT

	0 – 15
EVEN REGISTER	MORE SIGNIFICANT WORD
ODD REGISTER	LESS SIGNIFICANT WORD

To be an operand for a doubleword instruction which operates on register contents, the more significant word must be stored in a register location having an even address and the less significant word must be stored in the next higher (odd) register.

Many instruction and peripheral devices operate on eight-bit bytes which are packed two per register or memory location in the format:

BYTE DESIGNATIONS

BYTE 0	BYTE 1
0 – 7	8 – 15

Hexadecimal (base 16) digits are often used as a convenient means of representing binary byte, word or double word values. The hexadecimal word format is:

HEXADECIMAL DIGIT DESIGNATIONS

H_0	H_1	H_2	H_3
0 – 3	4 – 7	8 – 11	12 – 15

Where H_0 is the most significant digit in the word.

Where H_0 is the most significant digit in the word. Hexadecimal numbers and the equivalent decimal numbers are listed in Appendix A. In the text hexadecimal numbers appear in the form N_{16}.

Arithmetic Data Formats

Fixed Point Binary Integer Format - This is the standard arithmetic data format in the MODCOMP II and consists of a sign bit and 15 or 31 data bits. The most significant bit is the sign bit, which is defined:

Sign = 0, Positive or Zero Quantity
Sign = 1, Negative Quantity

Two's complement representation is used for negative numbers. The principal fixed point arithmetic formats are:

SINGLE PRECISION FIXED POINT DATA FORMAT

S	2^{14}	2^0
0	1	15

DOUBLE PRECISION FIXED POINT DATA FORMAT

S	2^{30}	2^{16}
0	1	15

2^{15}	2^0
0	15

Floating Point Format - This consists of a nine bit binary exponent and a 22 or 38 bit signed binary fraction. The exponent values are defined:

Exponent Value $_{16}$	Floating-Point Number	Value
000	2^{-256} X	Fraction Value $(1 > F \geq 0)$
400	2^0 X	Fraction Value
7FC	2^{255} X	Fraction Value

The value of zero is represented by $00000\ldots0_{16}$. Hardware operations resulting in a zero fraction set the exponent to all zeroes. A negative number is represented as the integer two's complement of the absolute value so that integer compare and negate operations are valid with both fixed point and floating point operands.

The floating point formats are:

SINGLE PRECISION FLOATING POINT DATA FORMAT

S	EXPONENT	FRACTION
0	1 — 9	10 — 15

LEAST SIGNIFICANT BITS OF FRACTION
0 — 15

DOUBLE PRECISION FLOATING POINT DATA FORMAT

S	EXPONENT	FRACTION
0	1 — 9	10 — 15

FRACTION
0 — 15

LEAST SIGNIFICANT BITS OF FRACTION
0 — 15

Character Formats

The ASCII code is the standard character code in MODCOMP computers and peripherals. Appendix B contains the character code definitions.

REGISTER FILE

MODCOMP II contains 16 addressable registers. Fifteen are fast access flip-flop registers having general register capabilities. Operands can be transferred between any of these registers and any other register or any memory location. In addition, the execution of many instructions produces a result stored in one or more of the general registers. All 15 of the general registers may be used in short indexed operations and 7 may be used as index registers.

One of the 16 addressable registers is the 16-bit switch register located on the control panel. This register is provided as one means of communication between the operator and program.

The designations and dedicated functions of the sixteen addressable registers are:

REGISTER FILE DESIGNATIONS AND DEDICATED FUNCTIONS

LOWER GENERAL REGISTER FILE AND INDEX REGISTERS (R_1-R_7)

R0	Switch Reg.
R1	Base Reg.
R2	
R3	
R4	
R5	
R6	
R7	

UPPER GENERAL REGISTER FILE

R8
R9
R10
R11
R12
R13
R14
R15

Register R1 has a dedicated hardware function in addition to being a general register. In one of the modes of memory address generation, a displacement value contained in the instruction is added to the contents of register R1 to produce the effective memory address. Registers R1-R7 may be used as index registers. All registers R0-R15 may be used in short indexed operations.

The registers are designated by four-bit fields in the formats of instructions which invoke register operation. Typical register designations are shown in the following examples:

REGISTER-TO-REGISTER INSTRUCTION FORMAT

OP. CODE	Ra	Rb
0 7	8 11	12 15

INDEXED INSTRUCTION FORMAT

OP. CODE	Ra	I	Rxx
0 7	8 11	12	13 15

SHORT INDEXED INSTRUCTION FORMAT

OP. CODE	Ra	Rx
0 7	8 11	12 15

Ra - Specifies one operand register ($0 \leq a \leq 15$) and the destination register. The destination register should not be R0, the switch register, unless the operation result is to be discarded, which is sometimes convenient in conditional branch instructions.

Rb - Specifies the second operand register ($0 \leq b \leq 15$).

Rxx - Specifies the index register ($1 \leq xx \leq 7$).

Rx - Specifies the effective address register for short indexed instructions ($0 \leq x \leq 15$).

ADDRESSING

Memory Word Addressing

A total of seven memory addressing modes are provided in MODCOMP II instructions which operate on word operands. In each of these modes, a 16-bit effective word address (EWA) is produced in the central processing unit (CPU) and sent to the memory system along with a read or write request. The 16-bit contents of the location specified by the EWA are then either read from memory or replaced by the word transferred from the CPU. The 16-bit EWA provides a direct addressing range of 65,536 words.

The first four of the seven memory addressing modes are derived from this instruction format:

BASIC MEMORY ADDRESS FORMAT

OP. CODE	Ra	I	Rxx	
0 — 7	8 — 11	12	13 — 15	1st INSTRUCTION WORD

ADDRESS	
0 — 15	2nd INSTRUCTION WORD

Ra - Register Address

Rxx - Index Register Address ($1 \leq xx \leq 7$) where 0 = no indexing

I - Indirect Address Bit

Direct Address Mode - If Rxx = 0 and I = 0, the 16-bit address contained in the second instruction word becomes the EWA.

Indexed Address Mode - If Rxx ≠ 0 and I = 0, the contents of register Rxx are added to the 16-bit address contained in the second instruction word. The least significant 16 bits of the result become the EWA. The contents of the index register may be either positive or negative to produce either positive or negative displacement indexing.

EXAMPLE:

DISCARD		0000 0000 0001 0100	ADDRESS = 20
CARRY		1111 1111 1111 0110	INDEX = -10
	1	0000 0000 0000 1010	EWA = 10

The indexing operation does not increase instruction execution time.

Indirect Address Mode - If Rxx= 0 and I = 1, the 16-bit address contained in the second instruction word specifies the memory location which contains the EWA. The indirect address capability is single level. One cycle time is added to instruction execution time by the indirect address word fetch.

Indexed and Indirect Address Mode - If Rxx $\neq$ 0 and I = 1, the contents of register Rxx are added to the 16-bit address contained in the second instruction word. The resulting address then specifies the location of the EWA. One cycle time is added to instruction execution time.

Immediate Mode - This two word memory reference instruction accesses operands or stores operands in the second instruction word. The program register is advanced by two to skip this location. The instruction format is:

IMMEDIATE OPERAND FORMAT

OP. CODE	Ra		
0 7	8 11	12 15	1st INSTRUCTION WORD

IMMEDIATE OPERAND	
0 15	2nd INSTRUCTION WORD

Short Displaced Mode - This single-word memory reference instruction format is provided for processing lists of operands occupying 16 or less consecutive memory locations. The instruction format is:

SHORT DISPLACEMENT FORMAT

OP. CODE	Ra	DF
0 7	8 11	12 15

DF = Displacement Field ($0 \leq DF \leq 15$)

In this mode of addressing memory, the positive displacement quantity DF and the contents of register R1 are added, to generate the 16-bit EWA. The contents of R1 are not modified by the EWA computation:

$$(R1) + DF = EWA$$

The 16-bit contents of R1 specify the base location (lowest address) of the list stored in memory.

When Branch instructions are executed in the short displaced mode, the Program Register rather than register R1 is used as the base register.

Short Indexed Format - This single-word memory reference instruction enables the contents of any of the 16 addressable registers to become the EWA. The instruction format is:

SHORT INDEXED FORMAT

OP. CODE	Ra	Rx
0 7	8 11	12 15

, where Rx specifies the register which contains the EWA.

Byte Addressing

A byte may be addressed in any memory word with a special form of the short indexed format. In this case Rx specifies an even/odd pair of general registers.

BYTE ADDRESS FORMAT

OP. CODE	Ra	Rx	
0 – 7	8 – 11	12 – 15	INSTRUCTION

BASE WORD ADDRESS	
0 – 15	EVEN REGISTER

SIGNED BYTE DISPLACEMENT

S	DISPLACEMENT WORD ADDRESS	B	
0	1 – 14	15	ODD REGISTER

B = 0 Specifies the byte contained in bits 0-7, and

B = 1 Specifies the byte contained in bits 8-15 of the memory location specified by the EWA

The effective byte address EBA is obtained by adding the 16-bit base address to the signed byte displacement which is first shifted right one bit position. This produces an EWA which enables the accessing of the location containing the specified byte. The proper byte is then accessed from this location depending upon the state of B.

Bit Addressing

Any bit in memory can be addressed by the instruction format:

BIT ADDRESSING FORMAT

OP. CODE	BIT NO.	I	Rxx
0 – 7	8 – 11	12	– 15

WORD ADDRESS
0 – 15

BIT NO. = 0 to 15, where 0 specifies the bit at the most significant end of the word.

The register-to-register, short displaced and short indexed forms are also used with these instructions.

OP. CODE	R	BIT #
0 — 7	8 — 11	12 — 15

BIT IN REGISTER

OP. CODE	BIT #	Rx
0 — 7	8 — 11	12 — 15

BIT IN MEMORY SHORT INDEXED

OP. CODE	BIT #	DF
0 — 7	8 — 11	12 — 15

BIT IN MEMORY SHORT DISPLACED

DEDICATED MEMORY LOCATIONS

Table 2-1 shows the area of memory which is dedicated to interrupt linkages and input/output transfer parameters.

Memory Locations $_{16}$	Dedicated Function
00-1F	Bootstrap Loader (00-2D), (Overlaps Interrupt Locations 20-2D)
20-5F	Interrupt Entry and Return
60-6F	DMP Transfer Count
70-7F	DMP Transfer Address
80-BF	I/O Data Interrupt Entry
C0-FF	I/O Service Interrupt Entry

Table 2-1 Dedicated Memory Locations

SECOND MEMORY PORT

This optional feature provides a second memory port, or access path. Each port can have access to all 64K words (max.) contained in one MODCOMP II/20 or II/25. The CPU packaged with the memory is connected to the first (higher priority) port. The second port, connected to an external CPU can be connected to any combination of 16K word memory sections. (0-16K, 16-32K, 32-48K, 48-64K).

Each port can obtain a memory access in a different memory section simultaneously. If a simultaneous access is attempted in the same 16K section by both ports, the higher priority port will obtain the next cycle and the lower priority port the following cycle.

READ-ONLY MEMORY CONTROLLER

All standard MODCOMP II instructions are executed by a sequence of micro instructions stored in a 256 x 40 bit solid-state LSI read-only memory (ROM) module. In addition the operations performed by the Direct Memory Processor are controlled by ROM stored micro instructions. Micro instructions are executed at the rate of 3.75 million instructions per second, which corresponds to three ROM cycles per main memory cycle.

The ROM can be expanded on a CPU option plane to enable macro instructions to be added to the standard instruction set.

OPERATIONAL INTEGRITY FEATURES

Continuous checking is performed for the principal conditions, for which valid checks can be made, that can cause machine stoppage or abnormal program operation. The error signals are connected to interrupt levels, either as standard or optional features, to facilitate operation of the computer in real-time environments.

Memory Parity

A parity bit is stored in all memory word locations, when the memory parity checking feature is present. Each time a memory access is made, either for a byte or a word, the parity of both bytes in the word is generated or checked. If an error is detected, the execution of the instruction is aborted, further execution is halted, and the machine attempts to trap to the optional parity priority interrupt level. (Level 1) If the optional System Protect Feature is included in the computer, the parity error signal is connected to the interrupt level (Level 1). Since the instruction execution is aborted when the error is detected, the signal which interrupts the computer is classified as a trap, rather than an interrupt signal. (See Traps - Pg. 4-5) The parity error light is reset by the interrupt.

The parity error indicator is set whenever a parity error is detected and will remain on until a priority interrupt occurs or the machine is normalized.

Overflow

An overflow signal is generated in arithmetic operations if the result exceeds the capacity designated for the result. The specific overflow conditions are defined with the individual instruction descriptions. The general instruction types which can cause overflow are: Add, Subtract, Divide, Two's Complement, and Left Arithmetic Shift.

If an overflow occurs during the execution of one of these instructions, the overflow latch will be set regardless of its previous condition. A special machine instruction (TRO,R) is used to read the latch and reset it. Another instruction (GMR,R,O) may be used to set the overflow latch unconditionally. (Displayed in register #3D bit 0).

Carry Save

A carry save signal is generated in arithmetic operations if the result produces a carry. The general instruction types which can produce a carry are: Add, Subtract, Multiply, Divide, and Two's Complement.

If a carry occurs as a result of execution of one of these instructions, the carry save latch (Register #3D bit 15) will be set regardless of its previous state. Conversely, if no carry is generated, the latch is reset regardless of its previous state. A special instruction (TRO,R) is used to read the latch. Any GMR,R instruction will unconditionally reset the carry save latch.

Unimplemented and Call Instructions

Optional instructions such as floating point and custom macro op code groups are trapped in MODCOMP II computers not containing these options. The trap routine can execute all of these instructions as subroutines. Therefore, programs which contain these optional instructions can be executed in all MODCOMP II computers.

The trap level, which is present in all machines, is Level 4.

A special instruction Request Executive Service (REX) always generates the Unimplemented Instruction trap. This instruction is used for communication with the resident executive.

Undefined Instructions

All undefined instruction op codes and unassigned op codes will execute as No Operation (NO OP) instructions in 800 nsec.

Floating Point Overflow

Floating point overflow is a separate trap from Overflow (above). Floating point overflow/underflow will occur if the resultant exponent of a floating point operation cannot be expressed within the range of the nine (9) bit binary exponent field of the floating point format.

The floating point unit trap mechanism used to indicate an overflow or underflow condition is the same function as the CPU trap implementation. The trap mechanism terminates the normal FPU flow of events and does not allow any results to be transferred back to the CPU register file. Therefore, the original register operands are maintained in the CPU register file and may be interrogated for further overflow-underflow clarification.

The trap level, when present in the system, is Level 5.

Doubleword Operand Register Storage

Doubleword operands must be stored in register pairs in which the more significant word is stored in an even numbered register and the less significant word is stored in the next higher (odd register). The even register number must be used in the instruction to designate the doubleword. The use of an odd register number to designate doublewords will produce unspecified results except for multiplication operations. Refer to the descriptions of multiply instructions for more information.

Power Fail Safe/Auto Start

When the a-c power is turned on or off in MODCOMP II computers having the PFS/AS feature, an interrupt is generated which overrides all other machine conditions, except the Halt condition.

This level is always enabled. When power fails, a minimum of 200 execution cycles are available after the interrupt occurs. After this time interval, memory writing is disabled to insure that the magnetic states of all cores remain unchanged when the power is turned off. When power is applied to the system, memory writing is also inhibited until proper initial conditions have been established for operation. At this time an interrupt is generated which can be used for automatic program initialization if the Halt/Run switch is in the RUN position or the CP is locked.

The PFS/AS interrupt level is Level 0.

System Protect Feature

The MODCOMP II offers a hardware system protect option which may utilize either a triple boundary method or a single (MCIII compatible) boundary method to assign protected memory. The rules governing the allocation of memory are as follows:

Triple Protect Boundary Registers

Two nine bit and one eight bit hardware registers are provided with the option to allow the selection of the protection boundaries for the memory system. These registers allow the boundaries to be assigned at any 128 word increment from 128 to 64K words of memory for the lower protect boundary, from 64K to 128 words of memory for the upper protect boundary, and at any 256 word increment from 256 to 64K words of memory for the third protect boundary.

1. For the Lower Boundary:
   (LPR) x (128) + 128 = Lower Boundary

-Where LPR = Contents of the 9 bit Lower Protect Boundary Register.
All memory below the lower boundary is protected (or privileged).

2. For the Upper Boundary:
   (UPR) x (128) + 128 = Upper Boundary

-Where UPR = Contents of the 9 bit Upper Protect Boundary Register
All memory at or above the upper boundary but below the third boundary is protected (or privileged).

3. For the Third Boundary:
   (B3R) x (256) + 256 = Third Boundary

-Where B3R = Protect Boundary Register #3.
Therefore, all memory at and above the lower boundary but below the upper boundary or all memory at or above the third boundary is unprotected (or unprivileged).

Single Protect Boundary Register

One five bit hardware register is provided to allow selection of the protection boundary for the memory system. This register allows the boundary to be assigned at any 2K word increment from 2K to 64K words of memory.

(PBR) x (2K) + 2K = Protect Boundary
-Where PBR = Protect Boundary Register

All memory below the protect boundary is protected (or privileged) and all memory at and above the boundary is unprotected (or unprivileged).

This protect scheme is program compatible with the system used in the MCIII.

General Program Protect Characteristics

These options provide the capability to prevent a background program(s) (located outside of protected memory) from erroneously altering, executing or permanently inhibiting the foreground (or protected) program(s). In addition, the protect logic is connected to an optional priority level to notify the computer of any attempt on its part to violate the protect structure even though the attempt was aborted (except for branch violations) by the hardware. When the protect option is invoked, the protect logic traps the execution of I/O, Interrupt and Halt instructions if the program is operating in unprotected memory.

The Master Clear Console switch and Power Normalization signals always fill all three protect registers thereby yielding all of existent memory protected.

The System Protect Feature consists of two types of protection:

Memory Write Protection is included to prevent programs from modifying other resident programs. In MODCOMP II, two boundaries can be established by program control at any 128 word boundaries in memory. Programs stored between these boundaries cannot modify or branch into locations outside of these two boundaries. If an illegal attempt is made, a trap is generated at interrupt Level 2.

The Request Executive Service instruction is used for communication between programs in unprotected memory and the resident executive, which is located in protected memory.

Privileged Instruction Execution capability is provided to prevent unprotected programs from executing any input/output, protect status, interrupt instructions or the Halt instruction. A trap is generated at interrupt Level 2 if the execution of any privileged instruction is attempted.

The standard memory parity error is connected to interrupt Level 1, as part of the System Protect Feature. This grouping of integrity features is the result of monitor requirements and the physical grouping of the interrupts.

REAL-TIME CLOCK

The real-time clock, which is part of the Executive Features, produces an interrupt signal at five millisecond intervals. The real-time clock interrupt is Level 6.

III. INSTRUCTION SET

OVERVIEW

All MODCOMP II instructions are described in this chapter. The instructions are grouped in the functional classes:

- Load, Store and Transfer
- Arithmetic
- Floating Point
- Logical
- Shift
- Bit Manipulation
- Byte Manipulation
- Unconditional Branch
- Control
- Interrupt and Call
- Input/Output

The principal MODCOMP instruction formats are:

0 — 7	8 — 11	12 — 15
OP CODE	a	b

Single Word Format

0 — 7	8 — 11	12 — 15
OP CODE	a	b
IMMEDIATE OPERAND		

Immediate Operand Format

0 — 7	8 — 11	12	13 — 15
OP CODE	a	I	b
MEMORY ADDRESS			

Two Word Format

Where: a and b define operand registers, index registers, bit address within a word, displacement address (up to 16 locations) with respect to a base address, shift count, interrupt level or peripheral device address and I specifies indirect addressing.

The general format for the instruction description is:

MNEMONIC   INSTRUCTION NAME   EXECUTION TIME

0 — 3	4 — 7	8 — 11	12 — 15
OP	CODE	Ra	Rb

Execution Description

Affected:

The Mnemonic is a three or four letter representation of the instruction name.

The Instruction Name briefly describes the function performed by the execution of the instruction.

The Execution Time is maximum (not average or minimum) and includes access time.

The Operation Code value is shown as two hexadecimal digits. The two right digits contain binary coded register addresses in many instructions and other binary coded fields in other instructions, as described. In all instructions in which the contents of register Rb, either with or without manipulation, are transferred to register Ra, the two register addresses may be made the same to produce a single register operation. For example, the contents of a register can be one's complemented by making the Ra and Rb addresses equal in the instruction Transfer One's Complement Register to Register.

Many instructions contain a second and some a third instruction word used for 16-bit memory addresses or immediate operands. The address in the Program Register (PR) referenced in the description of these instructions is that of the first instruction word.

The Execution Description covers all program controlled functions performed in the computer which comprise the instruction execution. In addition, the contents of the Program Register are advanced to the first word of the next instruction.

The Affected line lists all general registers and memory cells in which the contents are modified as a result of the execution of the instruction. In addition, if the execution of the instruction can cause overflow, the word "overflow" is included in the listing.

The symbols and abbreviations used in the instruction descriptions are listed alphabetically in the following table.

B	-	Byte designator bit (0 = left byte, 1 = right byte)
DF	-	Displacement Field, which is used in the short displaced addressing mode and has the value range $0 \leq DF \leq 15$
EA	-	Effective memory address, which is the address that results after all specified address manipulation operations have been completed
I	-	Indirect address bit
PR	-	Program Register, which is a 16-bit register containing the current program location
Ra	-	General register Ra, which is the operand destination register for many instructions
Ra, RaV1	-	Doubleword consisting of the concatenated values stored in register Ra (more significant half) and register RaV1 (less significant half), where Ra is even numbered register*
Ra_n	-	Bit n of register Ra
Rb	-	General register Rb, which is the operand source register for many instructions
Rxx	-	General register Rxx, $(1 \leq xx \leq 7)$ is the index register for many instructions. When Rxx = 0, no index operation occurs
Rx	-	Effective address register for short indexed instructions $(0 \leq x \leq 15)$
S	-	Sign bit
μs	-	Microseconds
()	-	Contents of
→	-	Replace the contents of
+	-	Addition operator
-	-	Subtraction operator
x	-	Multiplication operator
÷	-	Division operator
Λ	-	Logical AND operator
V	-	Logical OR operator
Ⓥ	-	Logical Exclusive OR operator
$\overline{(\ )}$	-	Logical NOT (One's complement) operator*

TABLE 3-1 Symbols and Abbreviations

*Ra,RaV1 normally indicate an even/odd register pair, 4 and 5 for example. RaV1 indicates that a binary one is logically OR'ed with Ra (hex value) so it follows that Ra,RaV1 cannot describe an even/odd register pair. If Ra = 5 then RaV1 also = 5.

**The 'Contents of' symbol ( ) is shown merely to show the physical position of the overline and is not necessarily part of the NOT symbol.

LOAD, STORE AND TRANSFER INSTRUCTIONS

This instruction group provides the capability to transfer information from memory to the general register file (load), from the general register file to memory (store) and from register to register (transfer). Either a byte, word or file consisting of from one to eight words can be transferred by single instruction execution. The word transfer instruction set includes all seven memory addressing modes - direct, indexed, indirect, indirect and indexed, immediate, short displaced and short indexed.

LDM

LOAD REGISTER FROM MEMORY — 2.4 us

0–3	4–7	8–11	12	13–15
E	5	Ra	I	Rxx
ADDRESS FIELD (0–15)				

(EA) → Ra

The contents of the effective memory location replace the contents of register Ra.

Affected: Ra

LDI

LOAD REGISTER FROM MEMORY IMMEDIATE — 1.6 us

0–3	4–7	8–11	12–15
E	D	Ra	//////
IMMEDIATE OPERAND (0–15)			

((PR) + 1) → Ra

The contents of the second instruction word replace the contents of register Ra.
Affected: Ra

LDS

LOAD REGISTER FROM MEMORY SHORT DISPLACED — 1.87 μs

0–3	4–7	8–11	12–15
F	5	Ra	DF

((R1) + DF) → Ra

The contents of the memory location specified by the displacement field DF added to the contents of register R1 replace the contents of register Ra.

Affected: Ra

LDX LOAD REGISTER FROM MEMORY SHORT INDEXED 1.87 us

0 3	4 7	8 11	12 15
F	D	Ra	Rx

((Rx)) → Ra

The contents of the memory location specified by the contents of register Rx replace the contents of register Ra.

Affected: Ra

STM STORE REGISTER IN MEMORY 2.4 us

0 3	4 7	8 11	12	13 15
E	6	Ra	I	Rxx
ADDRESS FIELD				
0				15

(Ra) → EA

The contents of register Ra replace the contents of the effective memory location.

Affected: (EA)

STI STORE REGISTER IN MEMORY IMMEDIATE 1.6 us

0 3	4 7	8 11	12 15
E	E	Ra	
IMMEDIATE OPERAND			
0			15

(Ra) → (PR) + 1

The contents of register Ra replace the contents of the second instruction word.

Affected: ((PR) + 1)

STS STORE REGISTER IN MEMORY SHORT DISPLACED 1.87 us

0 3	4 7	8 11	12 15
F	6	Ra	DF

(Ra) → (R1) + DF

The contents of register Ra replace the contents of the memory location specified by the displacement field DF added to the contents of register R1.

Affected: (EA)

STX — STORE REGISTER IN MEMORY SHORT INDEXED

1.87 us

0 3	4 7	8 11	12 15
F	E	Ra	Rx

(Ra) → Rx

The contents of register Ra replace the contents of the memory location specified by the contents of register Rx.

Affected: (EA)

LBX — LOAD BYTE FROM MEMORY

2.13 us

0 3	4 7	8 11	12 15
A	E	Ra	Rx

(EBA) → Ra_{8-15}

0 → Ra_{0-7}

0 15
BASE WORD ADDRESS

REGISTER Rx

0	1 14	15
S	SIGNED WORD DISPLACEMENT	B

REGISTER Rx V 1

The contents of the effective byte location replace the right byte in register Ra. Zeroes replace the left byte in register Ra. Register Rx specifies an even/odd pair of general registers which contain the base word address and the signed byte displacement. The byte designator B specifies the byte within the memory word (0 = left, 1 = right).

Affected: Ra

Effective Byte Address Generation

Byte addressing is a special form of short indexed addressing. The effective byte address is generated by the addition of the base word address and the signed byte displacement which consists of the signed word displacement and a byte designator B. During the instruction execution the signed word displacement is right shifted by one bit position and is then added to the base word address to form an effective word address. The equation can be interpreted as: $EBA = Rx + \frac{Rx\ V\ 1}{2}$

B = 0 Specifies the byte contained in bits 0-7

B = 1 Specifies the byte contained in bits 8-15

SBX STORE BYTE IN MEMORY 3.2 us

0–3	4–7	8–11	12–15
A	F	Ra	Rx

$(Ra_{8-15}) \rightarrow EBA$

0–15
BASE WORD ADDRESS

REGISTER Rx

0	1–14	15
S	SIGNED WORD DISPLACEMENT	B

REGISTER Rx V 1

The right byte in register Ra replaces the contents of the effective byte location. The other byte in the memory word is not affected. The byte designator B specifies the byte within the memory (0 = left, 1 = right). See Effective Byte Address Generation under the description of the Load Byte From Memory instruction.

Affected: (EBA)

LFM LOAD FILE FROM MEMORY 4.0 us + .8 (R-1)

0–3	4–7	8–11	12	13–15
A	4	Ra	I	Rxx
ADDRESS FIELD				

$(EA) \rightarrow Ra$
$(EA+1) \rightarrow Ra+1$
$(EA+N) \rightarrow R7$ (If $a \leq 7$)
$(EA+N) \rightarrow R15$ (If $7 < a \leq 15$)

The contents of from one to eight consecutive memory location starting with the effective memory location replace the contents of register Ra through R7, if $a \leq 7$, or register Ra through R15, if $7 < a \leq 15$.

Affected: Ra through R7/15

LFS LOAD FILE FROM MEMORY SHORT DISPLACED 3.47 + .8 (R-1)

0–3	4–7	8–11	12–15
B	4	Ra	DF

$((R1) + DF) \rightarrow Ra$
$((R1) + DF + 1) \rightarrow Ra+1$
$((R1) + DF + N) \rightarrow R7$ (If $a \leq 7$)
$((R1) + DF + N) \rightarrow R15$ (If $7 < a \leq 15$)

The contents of from one to eight consecutive memory locations starting with the location specified by the displacement field DF added to the contents of register R1 replace the contents of registers Ra through R7, if $a \leq 7$, or register Ra through R15, if $7 < a \leq 15$.

Affected: Ra through R7/15

LFX

LOAD FILE FROM MEMORY SHORT INDEXED

3.47 + .8 (R-1)

0 3	4 7	8 11	12 15
B	C	Ra	Rx

((Rx)) → Ra
((Rx)+1) → Ra+1
((Rx)+N) → R7 (If a ≤ 7)
((Rx)+N) → R15 (If 7 < a ≤ 15)

The contents of from one to eight consecutive memory locations starting with the location specified by the contents of Rx replace the contents of registers Ra through R7, if a ≤ 7, or register Ra through R15, if 7 < a ≤ 15.

Affected: Ra through R7/15

SFM

STORE FILE IN MEMORY

4.0 us + .8 (R-1)

0 3	4 7	8 11	12	13 15
A	5	Ra	I	Rxx
ADDRESS FIELD (0 15)				

(Ra) → EA
(Ra+1) → EA+1
(R7) → EA+N (If a ≤ 7)
(R15) → EA+N (If 7 < a ≤ 15)

The contents of registers Ra through R7, if a ≤ 7, or registers Ra through R15, if 7 < a ≤ 15, replace the contents of from one to eight consecutive memory locations starting with the effective memory location.

Affected: (EA) ... (EA+N)

SFS

STORE FILE IN MEMORY SHORT DISPLACED

3.47 us + .8 (R-1)

0 3	4 7	8 11	12 15
B	5	Ra	DF

(Ra) → (R1)+DF
(Ra+1) → (R1)+DF+1
(R7) → (R1)+DF+N (If a ≤ 7)
(R15) → (R1)+DF+N (If 7< a ≤15)

The contents of registers Ra through R7, if a ≤ 7, or registers Ra through R15, if 7 < a ≤ 15, replace the contents of from one to eight consecutive memory locations starting with the location specified by the displacement field DF added to the contents of register R1.

Affected: (EA)...(EA+N)

SFX

STORE FILE IN MEMORY SHORT INDEXED

3.47 us + .8 (R-1)

0 3	4 7	8 11	12 15
B	D	Ra	Rx

(Ra) → (Rx)
(Ra+1) → (Rx)+1
(R7) → (Rx)+N (If a ≤ 7)
(R15) → (Rx)+N (If 7< a≤ 15)

The contents of registers Ra through R7, if a ≤ 7, or registers Ra through R15, if 7 < a ≤ 15, replace the contents of from one to eight consecutive memory locations starting with the location specified by the contents of Rx.

Affected: (EA)...(EA+N)

TRR — TRANSFER REGISTER TO REGISTER — 0.8 us

0 — 3	4 — 7	8 — 11	12 — 15
6	D	Ra	Rb

(Rb) → Ra

The contents of register Rb replace the contents of register Ra.

Affected: Ra

TRRB — TRANSFER REGISTER TO REGISTER AND BRANCH IF NONZERO — 1.6 us

0 — 3	4 — 7	8 — 11	12 — 15
7	D	Ra	Rb
ADDRESS FIELD (0 — 15)			

(Rb) → Ra
If Result $\neq 0$, EA → PR
If Result $=0$, (PR)+2 → PR

The contents of register Rb replace the contents of register Ra.

If the results are unequal to zero, a branch is executed to the effective word location. If the results equal zero, the next instruction in sequence is executed.

Affected: Ra

ARITHMETIC INSTRUCTIONS

This instruction group includes the add, double-precision add, subtract, multiply, divide, compare, and two's complement instructions.

All instructions assume fixed-point operands, which may be scaled at any bit position. The double-precision add and divide instructions assume doubleword operands. All other instructions assume word operands.

All instructions, except the multiply and compare, produce an overflow if the conditions described with each instruction are met.

The multiply/divide instructions are a compatible set. Not only is the relationship true: (A x B) ÷ A = B, but also the positioning of the operands and results are consistent. In multiply operations, if Ra specifies an even numbered general register, the doubleword product is then stored in the even-odd register pair consisting of Ra and RaV1. If Ra specifies an odd numbered register, the least significant 16 bits of the product replace the multiplier in Ra. In divide, the doubleword dividend must be stored in an even-odd register pair Ra and RaV1. The quotient is then stored in RaV1 and the remainder in Ra. Therefore, the multiplier and quotient occupy the same register positions, which simplifies computations.

The maximum values of the products for word operand pairs having all combinations of signs are:

Operand Signs	Maximum Operands		Maximum Product
(+ x +)	$(2^{15}-1) \times (2^{15}-1)$	=	$2^{30} - 2^{16} + 2^{0}$
(+ x -)	$(2^{15}-1) \times 2^{15}$	=	$2^{30} - 2^{15}$
(- x -)	$2^{15} \times 2^{15}$	=	2^{30}

-where minus full scale = $1000\ 0000\ 0000\ 0000_2 = 2^{15}$.

None of these numbers exceed the capacity of a doubleword and therefore overflow cannot occur.

In the divide operation, overflow will occur if the quotient exceeds 16 bits in length. Two checks are made by the overflow checking logic to determine if this error condition exists:

(1) The sign and most significant bit of the dividend are compared. They must be equal; otherwise overflow will occur.

(2) The dividend is shifted left one bit position and then the divisor is subtracted from the most significant half. Overflow will occur if the absolute magnitude of the most significant half of the shifted dividend is not less than the absolute magnitude of the divisor.

As a result of the overflow logic the absolute magnitude of the largest permissable dividend is $2^{30} - 2^{15} - 2^{0}$.

Divide scaling is described in Appendix D.

ADM

ADD MEMORY TO REGISTER — 2.4 us

0–3	4–7	8–11	12	13–15
E	0	Ra	I	Rxx
ADDRESS FIELD (0–15)				

(EA) + (Ra) → Ra

The contents of the effective memory location are algebraically added to the contents of register Ra. The result is stored in register Ra. An overflow occurs if both operands have like signs but the result has the opposite sign.
Affected: Ra, Overflow

ADI

ADD MEMORY TO REGISTER IMMEDIATE — 1.6 us

0–3	4–7	8–11	12–15
E	8	Ra	(hatched)
IMMEDIATE OPERAND (0 1 … 15)			

((PR)+1) + (Ra) → Ra

The contents of the second instruction word are algebraically added to the contents of register Ra. The result is stored in register Ra. An overflow occurs if both operands have like signs but the result has the opposite sign.
Affected: Ra, Overflow

ADS

ADD MEMORY TO REGISTER SHORT DISPLACED — 1.87 us

0–3	4–7	8–11	12–15
F	0	Ra	DF

((R1)+DF) + (Ra) → Ra

The contents of the memory location specified by the displacement field DF added to the contents of register R1 are algebraically added to the contents of register Ra. The result is stored in register Ra. An overflow occurs if both operands have like signs but the result has the opposite sign.
Affected: Ra, Overflow

ADX

ADD MEMORY TO REGISTER SHORT INDEXED — 1.87 us

0–3	4–7	8–11	12–15
F	8	Ra	Rx

((Rx)) + (Ra) → Ra

The contents of the memory location specified by the contents of register Rx are algebraically added to the contents of register Ra. The result is stored in register Ra. An overflow occurs if both operands have like signs but the result has the opposite sign.

Affected: Ra, Overflow

ADMM

ADD REGISTER TO MEMORY — 3.47 us

0–3	4–7	8–11	12	13–15
C	0	Ra	I	Rxx
ADDRESS FIELD (0–15)				

(Ra) + (EA) → EA

The contents of register Ra are algebraically added to the contents of the effective memory location. The result is stored in the effective memory location. An overflow occurs if both operands have like signs but the result has the opposite sign.

Affected: Overflow, (EA)

ADMB

ADD REGISTER TO MEMORY AND BRANCH IF NONZERO — 4.27 us - BRANCH

0–3	4–7	8–11	12	13–15
C	4	Ra	I	Rxx
OPERAND ADDRESS FIELD (0–15)				
BRANCH ADDRESS FIELD (0–15)				

(Ra) + (EA) → EA
If Result ≠0, ((PR)+2) → PR
If Result =0, (PR)+3 → PR

The contents of register Ra are algebraically added to the contents of the effective memory location. The result is stored in the effective memory location. If the result does not equal zero, a branch is executed to the location specified by the third instruction word. Only the direct address mode without indexing is performed for the branch address. If the result equals zero, the next instruction in sequence is executed. An overflow occurs if both operands have like signs but the result has the opposite sign.

Affected: Overflow, (EA)

ADSM

ADD REGISTER TO MEMORY SHORT DISPLACED — 2.67 us

0–3	4–7	8–11	12–15
D	0	Ra	DF

(Ra) + ((R1) + DF) → (R1) + DF

The contents of register Ra are algebraically added to the contents of the effective memory location specified by the displacement field DF added to the contents of register R1. The result is stored in the effective memory location. An overflow occurs if both operands have like signs but the result has the opposite sign.

Affected: Overflow , (EA)

ADSB

ADD REGISTER TO MEMORY SHORT DISPLACED AND BRANCH IF NONZERO — 3.47 us-BRANCH

0–3	4–7	8–11	12–15
D	4	Ra	DF
BRANCH ADDRESS FIELD (0–15)			

(Ra) + ((R1) + DF) → (R1) + DF
If Result ≠0, ((PR)+1) → PR
If Result =0, (PR)+2 → PR

The contents of register Ra are algebraically added to the contents of the effective memory location specified by the displacement field DF added to the contents of register R1. The result is stored in the effective memory location. If the result does not equal zero, a branch is executed to the memory location specified by the contents of the second instruction word. If the result equals zero, the next instruction in sequence is executed. An overflow occurs if both operands have like signs but the result has the opposite sign.

Affected: Overflow , (EA)

ADXM

ADD REGISTER TO MEMORY SHORT INDEXED — 2.67 us

0–3	4–7	8–11	12–15
D	8	Ra	Rx

(Ra) + ((Rx)) → (Rx)

The contents of register Ra are algebraically added to the contents of the effective memory location specified by the contents of register Rx. The result is stored in the effective memory location. An overflow occurs if both operands have like signs but the result has the opposite sign.

Affected: Overflow , (EA)

ADXB

ADD REGISTER TO MEMORY SHORT INDEXED AND BRANCH IF NONZERO

3.47 us-BRANCH

0 3	4 7	8 11	12 15
D	C	Ra	Rx
BRANCH ADDRESS FIELD (0–15)			

(Ra) + ((Rx)) → (Rx)
If Result $\neq 0$, ((PR)+1) → PR
If Result =0, (PR)+2 → PR

The contents of register Ra are algebraically added to the contents of the effective memory location specified by the contents of register Rx. The result is stored in the effective memory location. If the result does not equal zero, a branch is executed to the memory location specified by the contents of the second instruction word. If the result equals zero, the next instruction in sequence is executed. An overflow occurs if both operands have like signs but the result has the opposite sign.

Affected: Overflow , (EA)

ADR

ADD REGISTER TO REGISTER

0.8 us

0 3	4 7	8 11	12 15
6	8	Ra	Rb

(Rb) + (Ra) → Ra

The contents of register Rb are algebraically added to the contents of register Ra. The result is stored in register Ra. An overflow occurs if both operands have like signs but the result has the opposite sign.

Affected: Ra, Overflow

ADRB

ADD REGISTER TO REGISTER AND BRANCH IF NONZERO

1.6 us

0 3	4 7	8 11	12 15
7	8	Ra	Rb
ADDRESS FIELD (0–15)			

(Rb) + (Ra) → Ra
If Result $\neq 0$, EA → PR
If Result =0, (PR)+2 → PR

The contents of register Rb are algebraically added to the contents of register Ra. The result is stored in register Ra. If the result does not equal zero, a branch is executed to the effective word location. If the result equals zero, the next instruction in sequence is executed. An overflow occurs if both operands have like signs but the result has the opposite sign.

Affected: Ra, Overflow

DAR

DOUBLE PRECISION ADD REGISTER TO REGISTER — 2.13 us

0–3	4–7	8–11	12–15
2	2	Ra	Rb

(Rb, RbV1) + (Ra, RaV1) → Ra, RaV1

The contents of registers Rb and RbV1 (with register Rb containing the more significant half and register RbV1 containing the less significant half of a double precision data word) are algebraically added to the contents of registers Ra and RaV1 (with register Ra containing the more significant half and register RaV1 containing the less significant half of a double precision data word). The sum replaces the contents of registers Ra and RaV1. Ra and Rb must specify even-numbered general registers.

Affected: Ra, RaV1, Overflow

SUM

SUBTRACT MEMORY FROM REGISTER — 2.4 us

0–3	4–7	8–11	12	13–15
E	1	Ra	I	Rxx
ADDRESS FIELD (0–15)				

(Ra) - (EA) → Ra

The contents of the effective memory location are algebraically subtracted from the contents of register Ra. The result is stored in register Ra. An overflow occurs when the result is the same as the sign of the subtrahend but is different from the sign of the minuend.

Affected: Ra, Overflow

SUI

SUBTRACT MEMORY FROM REGISTER IMMEDIATE — 1.6 us

0–3	4–7	8–11	12–15
E	9	Ra	////
IMMEDIATE OPERAND (0–15)			

(Ra) - ((PR)+1) → Ra

The contents of the second instruction word are algebraically subtracted from the contents of register Ra. The result is stored in register Ra. An overflow occurs when the result is the same as the sign of the subtrahend but is different from the sign of the minuend.

Affected: Ra, Overflow

SUS — SUBTRACT MEMORY FROM REGISTER SHORT DISPLACED — 1.87 us

0–3	4–7	8–11	12–15
F	1	Ra	DF

(Ra) - ((R1) +DF) → Ra

The contents of the memory location specified by the displacement field added to the contents of register R1 are algebraically subtracted from the contents of register Ra. The result is stored in register Ra. An overflow occurs if the sign of the result is the same as the sign of the subtrahend but is different from the sign of the minuend.

Affected: Ra, Overflow

SUX — SUBTRACT MEMORY FROM REGISTER SHORT INDEXED — 1.87 us

0–3	4–7	8–11	12–15
F	9	Ra	Rx

(Ra) - ((Rx)) → Ra

The contents of the memory location specified by the contents of register Rx are algebraically subtracted from the contents of register Ra. The result is stored in register Ra. An overflow occurs when the sign of the result is the same as the sign of the subtrahend but is different from the sign of the minuend.

Affected: Ra, Overflow

SUR — SUBTRACT REGISTER FROM REGISTER — 0.8 us

0–3	4–7	8–11	12–15
6	9	Ra	Rb

(Ra) - (Rb) → Ra

The contents of register Rb are algebraically subtracted from the contents of register Ra. The result is stored in register Ra. An overflow occurs when the sign of the result is the same as the sign of the subtrahend but is different from the sign of the minuend.

Affected: Ra, Overflow

SURB — SUBTRACT REGISTER FROM REGISTER AND BRANCH IF NONZERO — 1.6 us

0–3	4–7	8–11	12–15
7	9	Ra	Rb
ADDRESS FIELD (0–15)			

(Ra) - (Rb) → Ra
If Result ≠0, EA → PR
If Result =0, (PR)+2 → PR

The contents of register Rb are algebraically subtracted from the contents of register Ra. The result is stored in register Ra. If the result does not equal zero, a branch is executed to the effective word location. If the result equals zero, the next

instruction in sequence is executed. An overflow occurs when the sign of the result is the same as the sign of the subtrahend but is different from the sign of the minuend.

Affected: Ra, Overflow

MPM MULTIPLY MEMORY BY REGISTER 7.74 us

0–3	4–7	8–11	12	13–15
A	0	Ra	I	Rxx
ADDRESS FIELD (0–15)				

(EA) x (RAV1) → Ra, RaV1

The contents of the effective memory location (multiplicand) are multiplied by the contents of register RaV1 (multiplier). Ra normally specifies an even register so that the more significant half of the product replaces the contents of register Ra and the less significant half of the product replaces the contents of register RaV1. The sign of the product replaces the sign bit of register Ra. If Ra specifies an odd numbered register, the least significant 16 bits of the product replace the contents of register Ra.

Affected: Ra, RaV1

MPS MULTIPLY MEMORY BY REGISTER SHORT DISPLACED 7.21 us

0–3	4–7	8–11	12–15
B	0	Ra	DF

((R1)+DF) x (RaV1) → Ra, RaV1

The contents of the memory location specified by the displacement field DF added to the contents of register R1 (multiplicand) are multiplied by the contents of register RaV1(multiplier). Ra normally specifies an even register so that the more significant half of the product replaces the contents of register Ra and the less significant half of the product replaces the contents of register RaV1. The sign of the product replaces the sign bit of register Ra. If Ra specifies an odd numbered register, the least significant 16 bits of the product replace the contents of register Ra.

Affected: Ra, RaV1

MPX MULTIPLY MEMORY BY REGISTER SHORT INDEXED 7.21 us

0–3	4–7	8–11	12–15
B	8	Ra	Rx

((Rx)) x (RaV1) → Ra, RaV1

The contents of the memory location specified by the contents of Rx (multiplicand) are multiplied by the contents of register RaV1(multiplier). Ra normally specifies an even numbered register so that the product replaces the contents of register Ra and the less significant half of the product replaces the contents of register RaV1. The sign of the

product replaces the sign bit of register Ra. If Ra specifies an odd numbered register, the least significant 16 bits of the product replace the contents of register Ra.

Affected: Ra, RaVl

MPR

MULTIPLY REGISTER BY REGISTER

6.67 us

0–3	4–7	8–11	12–15
2	0	Ra	Rb

(Rb) x (RaVl) → Ra, RaVl

The contents of register Rb (multiplicand) are multiplied by the contents or register RaVl (multiplier). Ra normally specifies an even numbered register so that the more significant half of the product replaces the contents of register Ra and the less significant half of the product replaces the contents of register RaVl. The sign of the product replaces the sign bit of register Ra. If Ra specifies an odd numbered register, the least significant 16 bits of the product replace the contents of Ra.

Affected: Ra, RaVl

DVM

DIVIDE REGISTER BY MEMORY

12.2 us

0–3	4–7	8–11	12	13–15
A	1	Ra	I	Rxx
ADDRESS FIELD (0–15)				

(Ra, RaVl) ÷ (EA) → Ra, RaVl

The contents of the effective memory location (divisor) are divided into the contents of registers Ra and RaVl (dividend). The quotient replaces the contents of register RaVl and the remainder replaces the contents of register Ra. The sign of the quotient replaces the sign bit of register RaVl. Ra must specify an even numbered register. Overflow will occur if the quotient exceeds 16 bits.

Affected: Ra, RaVl Overflow

DVS

DIVIDE REGISTER BY MEMORY SHORT DISPLACED

11.4 us

0–3	4–7	8–11	12–15
B	1	Ra	DF

(Ra, RaVl) ÷ ((Rl)+DF) → Ra, RaVl

The contents of the memory location specified by the displacement field DF added to the contents of register Rl (divisor) are divided into the contents of registers Ra and RaVl (dividend). The quotient replaces the contents of register RaVl and the remainder replaces the contents of register Ra. The sign of the quotient replaces the

sign bit of register RaVl. Ra must specify an even numbered register. Overflow will occur if the quotient exceeds 16 bits.
Affected: Ra, RaVl Overflow

DVX

DIVIDE REGISTER BY MEMORY SHORT INDEXED 11.4 us

0 3	4 7	8 11	12 15
B	9	Ra	Rx

(Ra, RaVl) ÷ ((Rx)) → Ra, RaVl

The contents of the memory location specified by the contents of register Rx (divisor) are divided into the contents of registers Ra and Ravl (dividend). The quotient replaces the contents of register RaVl and the remainder replaces the contents of register Ra. The sign of the quotient replaces the sign bit of register RaVl. Ra must specify an even numbered register. Overflow will occur if the quotient exceeds 16 bits.
Affected: Ra, RaVl Overflow

DVR

DIVIDE REGISTER BY REGISTER 11.Ø us

0 3	4 7	8 11	12 15
2	1	Ra	Rb

(Ra, RaVl) ÷ (Rb) → Ra, RaVl

The contents of register Rb (divisor) are divided into the contents of registers Ra and RaVl (dividend). The quotient replaces the contents of register RaVl and the remainder replaces the contents of register Ra. The sign of the quotient replaces the sign bit of register RaVl. Ra must specify an even numbered register. Overflow will occur if the quotient exceeds 16 bits.
Affected: Ra, RaVl Overflow

CRMB

COMPARE MEMORY AND REGISTER 4.53 us

0 3	4 7	8 11	12	13 15
C	7	Ra	I	Rxx
OPERAND ADDRESS FIELD				
BRANCH ADDRESS FIELD (Ra) = (EA)				
BRANCH ADDRESS FIELD (Ra) < (EA)				

If (Ra) - (EA) =0, ((PR)+2) → PR
If (Ra) - (EA) < 0, ((PR)+3) → PR
If (Ra) - (EA) > 0, (PR)+4 → PR

The contents of the effective memory location are algebraically subtracted from the contents of register Ra. If the result equals zero, a branch is executed to the location specified by the third instruction word. If the result is negative, a branch is executed to the location specified by the fourth instruction word. Only the direct addressing mode without indexing is permitted for the branch operation. If the result is greater than zero, the next instruction in sequence is executed.
Affected: None

CRSB

COMPARE MEMORY AND REGISTER SHORT DISPLACED — 4.0 us

0 — 3	4 — 7	8 — 11	12 — 15
D	7	Ra	DF
BRANCH ADDRESS FIELD (EA)=(Ra)			
BRANCH ADDRESS FIELD (EA)>(Ra)			

0 — 15

If (Ra)-((R1)+DF)=0, ((PR)+1)→PR
If (Ra)-((R1)+DF)<0, ((PR)+2)→PR
If (Ra)-((R1)+DF)>0, (PR)+3→PR

The contents of the memory location specified by the displacement field added to the contents of register R1 are algebraically subtracted from the contents of register Ra. If the result equals zero, a branch is executed to the location specified by the second instruction word. If the result is negative, a branch is executed to the location specified by the third instruction word. Only the direct addressing mode without indexing is permitted for the branch operation. If the result is greater than zero, the next instruction in sequence is executed.
Affected: None

CRXB

COMPARE MEMORY AND REGISTER SHORT INDEXED — 4.0 us

0 — 3	4 — 7	8 — 11	12 — 15
D	F	Ra	Rx
BRANCH ADDRESS FIELD (EA)=(Ra)			
BRANCH ADDRESS FIELD (EA)>(Ra)			

0 1 — 15

If (Ra)-((Rx))=0, ((PR)+1)→PR
If (Ra)-((Rx))<0, ((PR)+2)→PR
If (Ra)-((Rx))>0, (PR)+3→PR

The contents of the memory location specified by the contents of register Rx are algebraically subtracted from the contents of register Ra. If the result equals zero, a branch is executed to the location specified by the second instruction word. If the result is negative, a branch is executed to the location specified by the third instruction word. Only the direct addressing mode without indexing is permitted for the branch operation. If the result is greater than zero, the next instruction in sequence is executed.
Affected: None

TRO

TRANSFER AND RESET OVERFLOW STATUS — 0.8 us

0 — 3	4 — 7	8 — 11	12 — 15
0	E	Ra	

(OVERFLOW) → Ra_0

(CARRY SAVE) → Ra_{15}

0 → OVERFLOW, Ra_{1-14}

The content of the overflow latch is transferred into the most significant bit of Ra and the last adder carry out saved is transferred into the least significant bit of Ra. Bits 1-14 of register Ra are set to zero and the overflow latch is reset by the execution of this instruction.
Affected: Ra, Overflow

TTR

TRANSFER TWO'S COMPLEMENT REGISTER TO REGISTER — 0.8 us

0 – 3	4 – 7	8 – 11	12 – 15
6	F	Ra	Rb

$\overline{(Rb)}+1 \rightarrow Ra$

The contents of register Ra are replaced by the two's complement of register Rb. An overflow occurs if the operand is minus full scale.

Affected: Ra, Overflow

TTRB

TRANSFER TWO'S COMPLEMENT REGISTER TO REGISTER AND BRANCH IF NONZERO — 1.6 us

0 – 3	4 – 7	8 – 11	12 – 15
7	F	Ra	Rb
ADDRESS FIELD (0 – 15)			

$\overline{(Rb)}+1 \rightarrow Ra$
If Result $\neq 0$, EA $\rightarrow$ PR
If Result $=0$, (PR)+2 $\rightarrow$ PR

The contents of register Ra are replaced by the two's complement of the contents of register Rb. If the result does not equal zero, a branch is executed to the effective word location. If the result equals zero, the next instruction in sequence is executed. An overflow occurs if the operand is minus full scale.

Affected: Ra, Overflow

LOGICAL INSTRUCTIONS

This group consists of the Extract $(\overline{A} \wedge B)$, OR $(A \vee B)$, Exclusive OR $(A \circledvee B)$, One's Complement, and Test instructions. All of these instructions operate on 16-bit operands. They produce a logical product (Extract), sum (OR), modulo-two sum (Exclusive OR), or complement and all but the Test instructions store the result in a general register or memory location. The Test instructions enable a comparison to be made between two operands without modifying either.

ETM

EXTRACT MEMORY FROM REGISTER 2.4 us

0 – 3	4 – 7	8 – 11	12	13 – 15
E	2	Ra	I	Rxx
ADDRESS FIELD				

$\overline{(EA)} \wedge (Ra) \rightarrow Ra$

The one's complement of the contents of the effective memory location are logically multiplied (AND function) by the contents of register Ra. The result is stored in register Ra.
Affected: Ra

ETI

EXTRACT MEMORY FROM REGISTER IMMEDIATE 1.6 us

0 – 3	4 – 7	8 – 11	12 – 15
E	A	Ra	
IMMEDIATE OPERAND (0 – 15)			

$\overline{((PR)+1)} \wedge (Ra) \rightarrow Ra$

The one's complement of the contents of the second instruction word are logically multiplied (AND function) by the contents of register Ra. The result is stored in register Ra.
Affected: Ra

ETS

EXTRACT MEMORY FROM REGISTER SHORT DISPLACED 1.87 us

0 – 3	4 – 7	8 – 11	12 – 15
F	2	Ra	DF

$\overline{((R1)+DF)} \wedge (Ra) \rightarrow Ra$

The one's complement of the contents of the memory location specified by the displacement field DF added to the contents of register R1 are logically multiplied (AND function) by the contents of register Ra. The result is stored in register Ra.
Affected: Ra

ETX

EXTRACT MEMORY FROM REGISTER SHORT INDEXED — 1.87 us

0–3	4–7	8–11	12–15
F	A	Ra	Rx

$\overline{((Rx))} \wedge (Ra) \rightarrow Ra$

The one's complement of the contents of the memory location specified by the contents of register Rx are logically multiplied (AND function) by the contents of register Ra. The result is stored in register Ra.

Affected: Ra

ETMM (5)

EXTRACT REGISTER FROM MEMORY — 3.47 us

0–3	4–7	8–11	12	13–15
C	1	Ra	I	Rxx
ADDRESS FIELD				

$\overline{(Ra)} \wedge (EA) \rightarrow EA$

The one's complement of the contents of register Ra are logically multiplied (AND function) by the contents of the effective memory location. The result is stored in the effective memory location.

Affected: (EA)

ETMB

EXTRACT REGISTER FROM MEMORY AND BRANCH IF NONZERO — 4.27 us-NO BRANCH

0–3	4–7	8–11	12	13–15
C	5	Ra	I	Rxx
OPERAND ADDRESS FIELD				
BRANCH ADDRESS FIELD				

$\overline{(Ra)} \wedge (EA) \rightarrow EA$
If Result $\neq 0$, $((PR)+2) \rightarrow PR$
If Result $=0$, $(PR)+3 \rightarrow PR$

The one's complement of the contents of register Ra are logically multiplied (AND function) by the contents of the effective memory location. The result is stored in the effective memory location. If the result does not equal zero, a branch is executed to the location specified by the third instruction word. Only the direct address mode without indexing is performed. If the result equals zero, the next instruction in sequence is executed.

Affected: (EA)

ETSM

EXTRACT REGISTER FROM MEMORY SHORT DISPLACED 2.67 us

0 3	4 7	8 11	12 15
D	1	Ra	DF

$\overline{(Ra)} \wedge ((R1)+DF) \rightarrow (R1)+DF$

The one's complement of the contents of register Ra are logically multiplied (AND function) by the contents of the effective memory location specified by the displacement field DF added to the contents of register R1. The result is stored in the effective memory location.
Affected: (EA)

ETSB

EXTRACT REGISTER FROM MEMORY SHORT DISPLACED AND BRANCH IF NONZERO 3.47 us

0 3	4 7	8 11	12 15
D	5	Ra	DF
BRANCH ADDRESS FIELD (0–15)			

$(Ra) \wedge ((R1)+DF) \rightarrow (R1)+DF$
If Result $\neq 0$ $((PR)+1) \rightarrow PR$
If Result $=0$ $(PR)+2 \rightarrow PR$

The one's complement of the contents of register Ra are logically multiplied (AND function) by the contents of the effective memory location specified by the displacement field DF added to the contents of register R1. The result is stored in the effective memory location. If the result does not equal zero, a branch is executed to the location specified by the second instruction word. Only the direct address mode without indexing is performed. If the result equals zero, the next instruction in sequence is executed.
Affected: (EA)

ETXM

EXTRACT REGISTER FROM MEMORY SHORT INDEXED 2.67 us

0 3	4 7	8 11	12 15
D	9	Ra	Rx

$\overline{(Ra)} \wedge ((Rx)) \rightarrow (Rx)$

The one's complement of the contents of register Ra are logically multiplied (AND function) by the contents of the effective memory location specified by the contents of register Rx. The result is stored in the effective memory location.
Affected: (EA)

ETXB

EXTRACT REGISTER FROM MEMORY SHORT INDEXED AND BRANCH IF NONZERO — 3.47 us

0 3	4 7	8 11	12 15
D	D	Ra	Rx
BRANCH ADDRESS FIELD (0–15)			

$\overline{(Ra)} \wedge ((Rx)) \rightarrow (Rx)$
If Result $\neq 0$ $((PR)+1) \rightarrow PR$
If Result $=0$ $(PR)+2 \rightarrow PR$

The one's complement of the contents of register Ra are logically multiplied (AND function) by the contents of the effective memory location specified by the contents of register Rx. The result is stored in the effective memory location. If the result does not equal zero, a branch is executed to the location specified by the second instruction word. Only the direct address mode without indexing is performed. If the result equals zero, the next instruction in sequence is executed.

Affected: (EA)

ETR

EXTRACT REGISTER FROM REGISTER — 0.8 us

0 3	4 7	8 11	12 15
6	A	Ra	Rb

$\overline{(Rb)} \wedge (Ra) \rightarrow Ra$

The one's complement of the contents of register Rb are logically multiplied (AND function) by the contents of register Ra. The result is stored in register Ra.

Affected: Ra

ETRB

EXTRACT REGISTER FROM REGISTER AND BRANCH IF NONZERO — 1.6 us

0 3	4 7	8 11	12 15
7	A	Ra	Rb
ADDRESS FIELD (0–15)			

$\overline{(Rb)} \wedge (Ra) \rightarrow Ra$
If Result $\neq 0$, $EA \rightarrow PR$
If Result $=0$, $(PR)+2 \rightarrow PR$

The one's complement of the contents of register Rb are logically multiplied (AND function) by the contents of register Ra. The result is stored in register Ra.

If the result does not equal zero, a branch is executed to the effective word location. If the result equals zero, the next instruction in sequence is executed.

Affected: Ra

ORM

OR MEMORY AND REGISTER

2.4 us

0–3	4–7	8–11	12	13–15
E	3	Ra	I	Rxx
ADDRESS FIELD				

(EA) V (Ra) → Ra

The contents of the effective memory location are logically added (OR function) to the contents of register Ra. The result is stored in register Ra.
Affected: Ra

ORI

OR MEMORY AND REGISTER IMMEDIATE

1.6 us

0–3	4–7	8–11	12–15
E	B	Ra	
IMMEDIATE OPERAND (0 1 … 15)			

((PR)+1) V (Ra) → Ra

The contents of the second instruction word are logically added (OR function) to the contents of register Ra. The result is stored in register Ra.
Affected: Ra

ORS

OR MEMORY AND REGISTER SHORT DISPLACED

1.87 us

0–3	4–7	8–11	12–15
F	3	Ra	DF

((R1)+DF) V (Ra) → Ra

The contents of the memory location specified by the displacement field DF added to the contents of register R1 are logically added (OR function) to the contents of register Ra. The result is stored in register Ra.
Affected: Ra

ORX

OR MEMORY AND REGISTER SHORT INDEXED

1.87 us

0–3	4–7	8–11	12–15
F	B	Ra	Rx

((Rx)) V (Ra) → Ra

The contents of the memory location specified by the contents of register Rx are logically added (OR function) to the contents of register Ra. The result is stored in register Ra.
Affected: Ra

ORMM

OR REGISTER AND MEMORY

3.47 us

0 – 3	4 – 7	8 – 11	12	13 – 15
C	2	Ra	I	Rxx
ADDRESS FIELD				

(Ra) V (EA) → EA

The contents of register Ra are logically added (OR function) to the contents of the effective memory location. The result is stored in the effective memory location.
Affected: (EA)

ORSM

OR REGISTER AND MEMORY SHORT DISPLACED

2.67 us

0 – 3	4 – 7	8 – 11	12 – 15
D	2	Ra	DF

(Ra) V ((R1)+DF) → (R1)+DF

The contents of register Ra are logically added (OR function) to the contents of the effective memory location specified by the displacement field DF added to the contents of register R1. The result is stored in the effective memory location.
Affected: (EA)

ORXM

OR REGISTER AND MEMORY SHORT INDEXED

2.67 us

0 – 3	4 – 7	8 – 11	12 – 15
D	A	Ra	Rx

(Ra) V ((Rx)) → (Rx)

The contents of register Ra are logically added (OR function) to the contents of the effective memory location specified by the contents of register Rx. The result is stored in the effective memory location.
Affected: (EA)

ORR

OR REGISTER AND REGISTER

0.8 us

0 – 3	4 – 7	8 – 11	12 – 15
6	B	Ra	Rb

(Rb) V (Ra) → Ra

The contents of register Rb are logically added (OR function) to the contents of register Ra. The result is stored in register Ra.
Affected: Ra

ORRB

OR REGISTER AND REGISTER AND BRANCH IF NONZERO — 1.6 us

0–3	4–7	8–11	12–15
7	B	Ra	Rb
ADDRESS FIELD (0–15)			

(Rb) V (Ra) → Ra
If Result ≠0, EA → PR
If Result =0, (PR)+2 → PR

The contents of register Rb are logically added (OR function) to the contents of register Ra. The result is stored in register Ra. If the result does not equal zero, a branch is executed to the effective word location. If the result equals zero, the next instruction in sequence is executed.

Affected: Ra

XOM

EXCLUSIVE OR MEMORY AND REGISTER — 2.4 us

0–3	4–7	8–11	12	13–15
E	4	Ra	I	Rxx
ADDRESS FIELD				

(EA) ⓥ (Ra) → Ra

The contents of the effective memory location are logically added modulo two (Exclusive Or function) to the contents of register Ra. The result is stored in register Ra.
Affected: Ra

XOI

EXCLUSIVE OR MEMORY AND REGISTER IMMEDIATE — 1.6 us

0–3	4–7	8–11	12–15
E	C	Ra	
IMMEDIATE OPERAND (0 1 … 15)			

((PR)+1) ⓥ (Ra) → Ra

The contents of the second instruction word are logically added modulo two (Exclusive Or function) to the contents of register Ra. The result is stored in register Ra.
Affected: Ra

XOS

EXCLUSIVE OR MEMORY AND REGISTER SHORT DISPLACED 1.87 us

0 3	4 7	8 11	12 15
F	4	Ra	DF

((R1)+DF) V (Ra) → Ra

The contents of the memory location specified by the displacement field DF added to the contents of register R1 are logically added modulo two (Exclusive Or function) to the contents of register Ra. The result is stored in register Ra.

Affected: Ra

XOX

EXCLUSIVE OR MEMORY AND REGISTER SHORT INDEXED 1.87 us

0 3	4 7	8 11	12 15
F	C	Ra	Rx

((Rx)) V (Ra) → Ra

The contents of the memory location specified by the contents of register Rx are logically added modulo two (Exclusive Or function) to the contents of register Ra. The result is stored in register Ra.

Affected: Ra

XOR

EXCLUSIVE OR REGISTER AND REGISTER 0.8 us

0 3	4 7	8 11	12 15
6	C	Ra	Rb

(Rb) V (Ra) → Ra

The contents of register Rb are logically added modulo two (Exclusive Or function) to the contents of register Ra. The result is stored in register Ra.

Affected: Ra

XORB

EXCLUSIVE OR REGISTER AND REGISTER AND BRANCH IF NONZERO 1.6 us

0 3	4 7	8 11	12 15
7	C	Ra	Rb
ADDRESS FIELD (0–15)			

(Rb) V (Ra) → Ra
If Result ≠0, EA → PR
If Result =0, (PR)+2 → PR

The contents of register Rb are logically added modulo two (Exclusive Or function) to the contents of register Ra. The result is stored in register Ra.

If the result does not equal zero, a branch is executed to the effective word location. If the result equals zero, the next instruction in sequence is executed.

Affected: Ra

TOR

TRANSFER ONE'S COMPLEMENT REGISTER TO REGISTER 0.8 us

0 3	4 7	8 11	12 15
0	D	Ra	Rb

$\overline{(Rb)} \rightarrow Ra$

The one's complement of the contents of register Rb replaces the contents of register Ra.
Affected: Ra

TRMB

TEST REGISTER AND MEMORY AND BRANCH IF ANY ONES COMPARE 3.73 us

0 3	4 7	8 11	12	13 15
C	6	Ra	I	Rxx
OPERAND ADDRESS FIELD				
BRANCH ADDRESS FIELD (0 1 … 15)				

$(Ra) \wedge (EA) \neq 0,\ ((PR)+2) \rightarrow PR$
$(Ra) \wedge (EA) = 0,\ (PR)+3 \rightarrow PR$

The contents of the effective memory location are logically multiplied (AND function) by the contents of register Ra. The result is not stored.

If the result does not equal zero, a branch is executed to the location specified by the third instruction word. Only the direct address mode without indexing is performed. If the result equals zero, the next instruction in sequence is executed.
Affected: None

TRSB

TEST REGISTER AND MEMORY SHORT DISPLACED AND BRANCH IF ANY ONES COMPARE 2.99 us

0 3	4 7	8 11	12 15
D	6	Ra	DF
BRANCH ADDRESS FIELD (0 … 15)			

$(Ra) \wedge ((R1)+DF) \neq 0,\ ((PR)+1) \rightarrow PR$
$(Ra) \wedge ((R1)+DF) = 0,\ (PR)+2 \rightarrow PR$

The contents of the memory location specified by the displacement field added to the contents of register R1 are logically multiplied (AND function) by the contents of register Ra. The result is not stored.

If the result does not equal zero, a branch is executed to location specified by the second instruction word. Only the direct address mode without indexing is performed. If the result equals zero, the next instruction in sequence is executed.
Affected: None

TRXB

TEST REGISTER AND MEMORY SHORT INDEXED AND BRANCH IF ANY ONES COMPARE

2.93 us

0 3	4 7	8 11	12 15
D	E	Ra	Rx
BRANCH ADDRESS FIELD (0–15)			

$(Ra) \wedge ((Rx)) \neq 0$, $((PR)+1) \rightarrow PR$
$(Ra) \wedge ((Rx)) = 0$, $(PR)+2 \rightarrow PR$

The contents of the memory location specified by the contents of register Rx are logically multiplied (AND function) by the contents of register Ra. The result is not stored.

If the result does not equal zero, a branch is executed to location specified by the second instruction word. Only the direct address mode without indexing is performed. If the result equals zero, the next instruction in sequence is executed.
Affected: None

TERB

TEST REGISTER AND REGISTER AND BRANCH IF ANY ONES COMPARE

1.6 us

0 3	4 7	8 11	12 15
7	E	Ra	Rb
ADDRESS FIELD (0–15)			

$(Rb) \wedge (Ra) \rightarrow$ RESULT
If Result $\neq 0$, EA $\rightarrow$ PR
If Result $=0$, $(PR)+2 \rightarrow PR$

The contents of register Rb are logically multiplied (AND function) by the contents of register Ra. The result is not stored.

If the result does not equal zero, a branch is executed to the effective word location. If the result equals zero, the next instruction in sequence is executed.

Affected: None

FLOATING POINT INSTRUCTIONS

INTRODUCTION

The optional floating point arithmetic instructions provide the capability to process very large or very small magnitude operands with precise results.

Floating point numbers consist of three parts: a sign, an exponent and a fraction. The sign bit applies only to the fraction. The exponent is a biased nine-bit binary number. The fraction is a binary number with an assumed radix point to the left of the high-order digit. The quantity that the floating-point number represents is obtained by raising the fraction value to the power expressed in the exponent value.

Data Formats

Floating point numbers are fixed in length and are either two word single precision or three word double precision in format.

The first bit (bit 0) in both formats is the sign of the fraction. A one (1) bit represents a minus sign and a zero bit (0) represents a positive sign. The next nine bits (2^1-2^{10}) represent a biased binary exponent. The fraction contains a 22 bit binary number (single-precision format) or a 38 bit binary number (double precision format).

The single precision format allows faster processing and uses less storage. The double precision format while providing greater precision, requires more processing time and use of an additional register and/or memory location.

Single Precision Floating Point Number

	EA/Ra/Rb		EA+1/RaV1/RbV1
S^1	$EXPONENT^9$	FRACTION	22
0	1 — 10	11 — 16	0 — 16

Double Precison Floating Point Number

	EA/Ra/Rb		EA+1/RaV1/RbV1	EA+2/RaV2/RbV2
S^1	$EXPONENT^9$		FRACTION	38
0	1 — 10	11 — 16	0 — 16	0 — 16

Ra OR Rb FIELD	SINGLE PRECISION OPERAND (OR RESULTS) REGISTERS USED	DOUBLE PRECISION OPERAND (OR RESULTS) REGISTERS USED
0	0, 1	0, 1, 2
1	1, 1	1, 1, 3
2	2, 3*	2, 3, 2
3	3, 3	3, 3, 3
4	4, 5*	4, 5, 6
5	5, 5	5, 5, 6
6	6, 7*	6, 7, 6
7	7, 7*	7, 7, 7
8	8, 9*	8, 9, A*
9	9, 9	9, 9, B
A	A, B*	A, B, A
B	B, B	B, B, B
C	C, D*	C, D, E*
D	D, D	D, D, F
E	E, F*	E, F, E
F	F, F	F, F, F

* Indicates all normally useful selections

TABLE 3-2 Floating Point Register Selections

FLOATING POINT INSTRUCTION MNEMONICS

This group of 16 optional instructions is made up of the four arithmetic operations; add, subtract, multiply and divide. Each of the four arithmetic operations can be executed in register-to-register or memory-to-register formats with either single precision or double precision operands.

Add	Subtract	Multiply	Divide
FAR	FSR	FMR	FDR Reg-Reg
FARD	FSRD	FMRD	FDRD R-R Double
FAM	FSM	FMM	FDM Mem-Reg
FAMD	FSMD	FMMD	FDMD M-Reg Double

GENERAL RULES

When floating point instructions specify Ra,RaVl and Rb,RbVl, Ra and Rb must specify even numbered registers which will contain the more significant half of a single precision floating point operand, and RaVl, RbVl will specify the next sequential odd numbered registers and hold the less significant half of a single precision floating point operand. Refer to Table 3-2 for normally useful register selections.

Operands presented to the Floating Point Unit will be in normalized form and likewise, operation results will always be normalized.

The exceptions to this rule are when an unnormalized number is presented to the Floating Point Unit for normalization (e.g. 0 + an unnormalized number will yield the same number in a normalized format), and when a zero fraction is used.

The storage of floating point operands, both in CPU registers and in memory, follow the same rules used for fixed point operands handled in the standard MODCOMP II. That is, the most significant word of the operands is stored in the lower memory location or lower general purpose register number.

Example:

	EWA Ra/Rb	EWA+1 RaV1	EWA+2 RaV2
Word	S (0) | EXPONENT (1–9) | MSB'S OF FRACTION (10–15)	FRACTION (0–15)	LEAST SIGNIFICANT BITS OF FRACTION (0–15)
FOR REGISTERS	R4	R5	R6
FOR MEMORY (EA)	X	X+1	X+2

Overflow

Floating point overflow/underflow occurs if the resultant exponent of a floating point operation cannot be expressed within the range of the nine bit binary exponent field of the floating point format.

A trap occurs at interrupt level 5 if floating point overflow/underflow is detected. See Sections II and IV for a detailed explanation of traps.

FAR FLOATING POINT ADD REG. TO REG.

0–3	4–7	8–11	12–15
3	0	Ra	Rb

15 us

(Rb),(RbV1)+(Ra),(RaV1)→Ra,RaV1

The contents of registers Rb and RbV1 (with register Rb containing the more significant half and register RbV1 containing the less significant half of a single precision floating point operand) are algebraically added to the contents of registers Ra and RaV1 (with register Ra containing the more significant half and register RaV1 containing the less significant half of a single precision floating point operand). The sum replaces the contents of registers Ra and RaV1. Ra and Rb must specify even-numbered registers. A floating point overflow will occur if the resultant exponent cannot be expressed within the range of the nine (9) bit binary exponent field of the floating point format.

Affected: Ra,RaV1

FSR FLOATING POINT SUBTRACT REG FROM REG

3	1	Ra	Rb

15 us

(Ra),(RaV1)-(Rb),(RbV1)→Ra,RaV1

The contents of registers Rb and RbV1 (with register Rb containing the more significant half and register RbV1 containing the less significant half of a single precision floating point operand) are algebraically subtracted from the contents of registers Ra and RaV1 (with register Ra containing the more significant half and register RaV1 containing the less significant half of a single precision floating point operand). The result is stored in registers Ra and RaV1. Ra and Rb must specify even numbered registers. A floating point overflow will occur if the resultant exponent cannot be expressed within the range of the nine (9) bit binary exponent field of the floating point format.

Affected: Ra,RaV1

FMR FLOATING POINT MULTIPLY REG BY REG

3	2	Ra	Rb

12.5 us

(Rb),(RbV1)x(Ra),(RaV1)→Ra,RaV1

The contents of registers Rb and RbV1 (with register Rb containing the more significant half and register RbV1 containing the less significant half of a single precision floating point multiplicand) are multiplied by the contents of registers Ra and RaV1 (with register Ra containing the more significant half and register RaV1 containing the less significant half of a single precision floating point multiplier). The product is stored in registers Ra and RaV1. Ra and Rb must specify even numbered registers. A floating point overflow will occur if the resultant exponent cannot be expressed within the range of the nine (9) bit binary exponent field of the floating point format.

Affected: Ra,RaV1

FDR FLOATING POINT DIVIDE REG BY REG

3	3	Ra	Rb

13 us

(Ra),(RaV1)÷(Rb),(RbV1)→Ra,RaV1

The contents of registers Rb and RbV1 (with register Rb containing the more significant half and register RbV1 containing the less significant half of a single precision floating point divisor) are divided into the contents of registers Ra and RaV1 (with register Ra containing the more significant half and register RaV1

containing the less significant half of a single precision floating point dividend). The quotient replaces the contents of registers Ra and RaV1. Ra and Rb must specify even-numbered registers. A floating point overflow will occur if the divisor is equal to zero or if the resultant exponent cannot be expressed within the range of the nine (9) bit binary exponent field of the floating point format.

Affected: Ra,RaV1

FARD FLOATING POINT ADD REG TO REG DOUBLE

3	4	Ra	Rb

20.5 us

(Rb),(RbV1),(RbV2)+(Ra),(RaV1),(RaV2)→Ra,Rav1,RaV2

The contents of registers Rb,RbV1, and RbV2 (with these registers arranged in significance as described in Table 3-2 under floating point double precision operand formats) are algebraically added to the contents of registers Ra,RaV1, and RaV2 (with these registers arranged in significance as described in Table 3-2 under floating point double precision operand formats). The sum replaces the contents of registers Ra,RaV1, and RaV2. Ra and Rb must specify general purpose register four (4_{16}), eight (8_{16}), or C_{16}. A floating point overflow will occur if the resultant exponent cannot be expressed within the range of the nine (9) bit binary exponent field of the floating point format.

Affected: Ra,RaV1,RaV2

FSRD FLOATING POINT SUBTRACT REG FROM REG DOUBLE

3	5	Ra	Rb

20.5 us

(Ra),(RaV1),(RaV2)-(Rb),(RbV1),(RbV2)→Ra,RaV1,RaV2

The contents of registers Rb,RbV1,RbV2 are algebraically subtracted from the contents of registers Ra,RaV1,RaV2. The result is stored in registers Ra,RaV1,RaV2. Ra and Rb must specify general purpose register four (4_{16}), eight (8_{16}) or (C_{16}). Floating point overflow may occur as described previously.

Affected: Ra,RaV1,RaV2

FMRD FLOATING POINT MULTIPLY REG BY REG DOUBLE

3	6	Ra	Rb

16 us

(Rb),(RbV1),(RbV2)x(Ra),(RaV1,(RaV2)→Ra,RaV1,RaV2

The contents of registers Rb,RbV1,RbV2 containing a double precision floating point multiplicand are multiplied by the contents of Ra,RaV1,RaV2 which hold the double precision floating point multiplier. The product is stored in registers Ra,RaV1, RaV2. Ra and Rb must specify general purpose registers four (4_{16}), eight (8_{16}) or (C_{16}). Floating point overflow may occur as described previously.

Affected: Ra,RaV1,RaV2

FDRD FLOATING POINT DIVIDE REG BY REG DOUBLE

3	7	Ra	Rb

16.5 us

(Ra),(RaV1),(RaV2)÷(Rb),(RbV1),(RbV2)→Ra,RaV1,RaV2

The contents of registers Rb,RbV1,RbV2 containing a double precision floating point divisor are divided into the contents of registers Ra,RaV1,RaV2 which hold the double precision floating point dividend. The quotient replaces the contents of registers Ra,RaV1,RaV2. Ra and Rb must specify general purpose registers four (4_{16}), eight (8_{16}) or (C_{16}). Floating Point overflow may occur as described previously.

Affected: Ra,RaV1,RaV2

FAM FLOATING POINT ADD MEMORY TO REGISTER

3	8	Ra	Rx
ADDRESS WORD			

17.5 us

(EA),(EA+1)+(Ra),(RaV1)→Ra,RaV1

The contents of the effective memory location and the effective memory location plus one (with (EA) containing the less significant half of a single precision floating point operand and (EA+1) containing the more significant half of a single precision floating point operand) are algebraically added to the contents of registers Ra and RaV1 (with register Ra containing the more significant half and register RaV1 containing the less significant half of a single precision floating point operand). The sum replaces the contents of Ra and RaV1. Ra must specify an even-numbered register. A floating point overflow will occur if the result exponent cannot be expressed within the range of the nine (9) bit binary exponent field of the floating point format.

Affected: Ra,RaV1

FSM FLOATING POINT SUBTRACT MEMORY FROM REGISTER

3	9	Ra	Rx
ADDRESS WORD			

17.5 us

(Ra),(RaV1)-(EA),(EA+1)→Ra,RaV1

The contents of the EA and EA+1 are algebraically subtracted from the contents of Ra,RaV1. The remainder replaces the contents of Ra,RaV1. Ra must specify an even numbered register.

Floating point overflow may occur as described previously.

Affected: Ra,RaV1

FMM FLOATING POINT MULTIPLY MEMORY BY REGISTER

3	A	Ra	Rx
ADDRESS WORD (EA)			

14.5 us

(EA), (EA+1)x(Ra),(RaV1)→Ra,RaV1

The contents of the EA and EA+1 containing a single precision fixed point multiplicand are multiplied by the contents of Ra,RaV1 containing a single precision floating point multiplier. The product replaces the contents of Ra,RaV1. Ra must specify an even numbered register.

Floating point overflow may occur as described previously.

Affected: Ra,RaV1

FDM FLOATING POINT DIVIDE MEMORY INTO REGISTER

3	B	Ra	Rx
ADDRESS WORD			

15.5

(Ra),(RaV1)÷(EA),(EA+1)→Ra,RaV1

The contents of Ra and RaV1 containing a single precision floating point dividend are divided by the contents of EA and EA+1 containing a single precision floating point divisor. The quotient replaces the contents of Ra,RaV1. Ra must specify an even numbered register.

Floating point overflow may occur as described previously.

Affected: Ra,RaV1

FAMD FLOATING POINT ADD MEMORY TO REGISTER DOUBLE

3	C	Ra	Rx
ADDRESS WORD			

22.5 us

(EA),(EA+1),(EA+2)+(Ra),(RaV1),(RaV2)→Ra,RaV1,RaV2

The contents of EA,EA+1,EA+2 containing a double precision floating point augend are algebraically added to the contents of Ra,RaV1,RaV2 containing a double precision floating point addend. The sum replaces the contents of Ra,RaV1,RaV2. Ra must specify general purpose register four (4_{16}), eight (8_{16}) or (C_{16}).

Floating point overflow may occur as described previously.

Affected: Ra,RaV1,RaV2

FSMD FLOATING POINT SUBTRACT MEMORY FROM REG DOUBLE

3	D	Ra	Rx
ADDRESS WORD			

22.5 us

(Ra),(RaV1),(RaV2)-(EA),(EA+1),(EA+2)→Ra,RaV1,RaV2

The contents of EA,EA+1 and EA+2 contain a double precision floating point subtrahend which is subtracted from Ra,RaV1,RaV2 containing a double precision floating point minuend. The remainder replaces the contents of Ra,RaV1,RaV2. Ra must specify general purpose register four (4_{16}), eight (8_{16}) or (C_{16}).

Floating point overflow may occur as described previously.

Affected: Ra,RaV1,RaV2

FMMD FLOATING POINT MULTIPLY MEMORY BY REGISTER DOUBLE

3	E	Ra	Rx
ADDRESS WORD			

18 us

(EA),(EA+1),(EA+2)x(Ra),(RaV1),(RaV2)→Ra,RaV1,RaV2

The double precision floating point multiplicand contained in EA,EA+1 and EA+2 is multiplied by the double precision floating point multiplier contained in registers Ra,RaV1 and RaV2. The product replaces the contents of Ra,RaV1 and RaV2. Ra must specify general purpose register four (4_{16}), eight (8_{16}), or (C_{16}).

Floating point overflow may occur as described previously.

Affected: Ra,RaV1,RaV2

FDMD FLOATING POINT DIVIDE MEMORY INTO REG DOUBLE

3	F	Ra	Rx
ADDRESS WORD			

19 us

(Ra),(RaV1),(RaV2)÷(EA),(EA+1),(EA+2)→Ra,RaV1,RaV2

The double precision floating point dividend contained in Ra,RaV1 and RaV2 is divided by the double precision floating point divisor contained in EA,EA+1 and EA+2. The quotient replaces the contents of Ra,RaV1 and RaV2. Register Ra must specify general purpose register four (4_{16}), eight (8_{16}) or (C_{16}).

Floating point overflow may occur as described previously.

Affected: Ra,RaV1,RaV2

SHIFT INSTRUCTIONS

The ten instructions in this group are used to reposition bits left or right within a single or a pair of adjacent registers. All combinations of arithmetic and logical, left and right, and single register and double register shift operations are provided. In addition, a left rotate instruction is included.

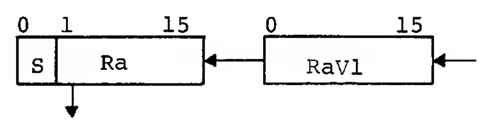
The execution of each shift instruction may shift the operand zero to 15 bit positions as defined by the binary coded shift field (bits 12-15) in each instruction except LRS.

In all double register shift operations, the register specified by the instruction word must be the even register of an even-odd register pair consisting of two adjacent registers in the general register file (Ra (even) and RaV1 (odd)). In doubleword arithmetic shifts, the more significant half of the operand is assumed to be in the even register and the less significant half in the odd register.

LAD

SHIFT LEFT ARITHMETIC DOUBLE — 1.87 us + 267(N-1) ns

0 3	4 7	8 11	12 15
2	E	Ra	SHIFTS

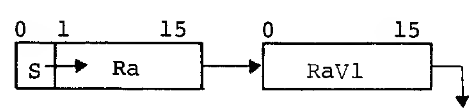


The contents of register Ra and register RaV1 are shifted left zero to 15 bit position(s) as specified by the shift count control field. The sign bit of Ra does not change either during or after the shift. Zeros are shifted into the LSB position of RaV1 and the MSB of RaV1 is shifted into the least significant bit position of Ra with each shift step. The next to MSB of Ra (bit position 1) is shifted out of the register and is lost. Ra must specify an even general register.
Affected: Ra, RaV1, Overflow

RAD

SHIFT RIGHT ARITHMETIC DOUBLE — 1.87 us + 267(N-1)ns

0 3	4 7	8 11	12 15
2	A	Ra	SHIFTS



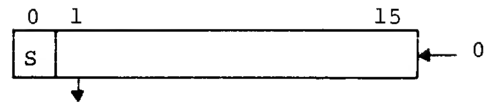
The contents of register Ra and register Ra+1 are shifted right zero to 15 bit position(s) as specified by the shift count control field. The sign bit of Ra does not change either during or after the shift.* The LSB(s) of Ra are shifted to the MSB position of RaV1 and the LSB(s) of RaV1 are shifted out of the register and are lost. Ra must specify an even general register.
Affected: Ra, RaV1

*The sign bit is propogated right the number of places specified by the shift count.

LAS

SHIFT LEFT ARITHMETIC SINGLE 1.6 us + 267(n-1)ns

0 3	4 7	8 11	12 15
2	F	Ra	SHIFTS

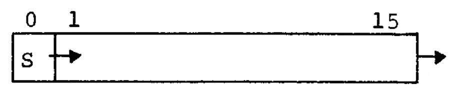


The contents of register Ra are shifted left zero to 15 bit position(s) as specified by the shift count control field. The sign bit of Ra does not change either during or after the shift. The next to MSB of Ra (bit position 1) is shifted out of the register and is lost. Zeros are shifted into the LSB position(s) of Ra.
Affected: Ra, (Overflow)

RAS

SHIFT RIGHT ARITHMETIC SINGLE 1.6 us + 267(N-1) ns

0 3	4 7	8 11	12 15
2	B	Ra	SHIFTS

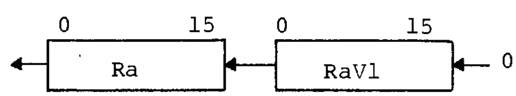


The contents of register Ra are shifted right zero to 15 bit position(s) as specified by the shift count control field. The sign bit of Ra does not change either during or after the shift* The least significant bit(s) of Ra are shifted out of the register and are lost.
Affected: Ra *See previous page.

LLD

SHIFT LEFT LOGICAL DOUBLE 1.87 us + 267(N-1) ns

0 3	4 7	8 11	12 15
2	C	Ra	SHIFTS

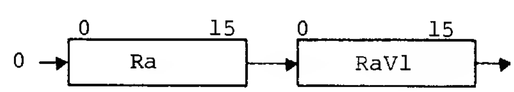


The contents of register Ra and register Ra+1 are shifted left zero to 15 bit position(s) as specified by the shift count control field. Zeros are shifted into the least significant bit position(s) of RaV1 and the most significant bit(s) of RaV1 are shifted into the least significant bit position(s) of Ra. The most significant bit(s) of Ra are shifted out of Ra and are lost. Ra must specify an even general register.
Affected: Ra, RaV1

RLD

SHIFT RIGHT LOGICAL DOUBLE 1.87 us + 267(N-1) ns

0 3	4 7	8 11	12 15
2	8	Ra	SHIFTS

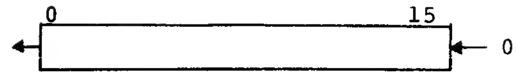


The contents of register Ra and register RaV1 are shifted right zero to 15 bit position(s) as specified by the shift count control field. Zeros are shifted into the most significant bit position(s) of Ra and the least significant bit position(s) of Ra are shifted into the most significant bit position(s) of RaV1. The least significant bit(s) of RaV1 are shifted out of RaV1 and are lost. Ra must specify an even general register.
Affected: Ra, RaV1

LLS SHIFT LEFT LOGICAL SINGLE

1.6 us + 267(N-1) ns

0 3	4 7	8 11	12 15
2	D	Ra	SHIFTS



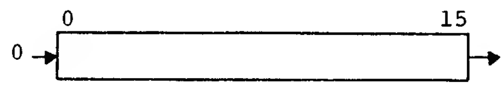
The contents of register Ra are shifted left zero to 15 bit position(s) as specified by the shift count control field. Zeros are shifted into the least significant bit position(s) and the most significant bit position(s) are shifted out of Ra_0 and are lost.

Affected: Ra

RLS SHIFT RIGHT LOGICAL SINGLE

1.6 us + 267(N-1) ns

0 3	4 7	8 11	12 15
2	9	Ra	SHIFTS



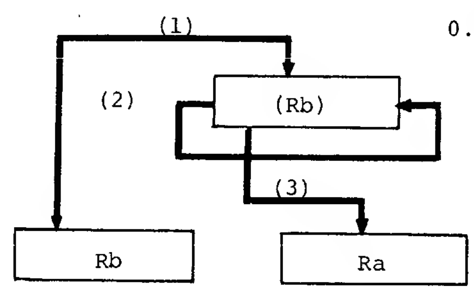
The contents of register Ra are shifted right zero to 15 bit positions as specified by the shift count control field. Zeros are shifted into the most significant bit position(s) and the least significant bit position(s) are shifted out of Ra_{15} and are lost.

Affected: Ra

LRS LEFT ROTATE SINGLE

0.8 us

0 3	4 7	8 11	12 15
0	F	Ra	Rb



$(Rb_{01-15}) \longrightarrow Ra_{00-14}$

$(Rb_{00}) \longrightarrow Ra_{15}$

The contents of register Ra are replaced by the contents of register Rb shifted left one bit position with the most significant bit of Rb rotated into bit position 15 of register Ra. The contents of Rb are unaffected.

Affected: Ra

BIT MANIPULATION INSTRUCTIONS

The bit manipulation instruction group includes the Load, Add, Subtract, Zero, OR, Exclusive OR, Test, and Compare instructions. In all instructions except Zero and Test, one operand is a bit literal of value one and the other operand is the 16-bit contents of the effective memory location or designated register. For example the Add Bit In Memory instruction causes a bit of value one to be added to the contents of the effective memory location. Carry is propagated through to the left through the sign bit.

The position of the bit literal is designated by the four bit, binary coded Bit Field in each instruction. Any bit in the word can be designated. The value of the Bit Field (n) specifies a 16-bit binary number of value $+2^{15-n}$.

Since the value of the bit literal is always one in the OR instruction, execution of this instruction causes the designated bit in memory or a general register to be set to one. For the same reason, execution of the Exclusive OR instruction causes the designated bit to be complemented (inverted).

LBR LOAD BIT IN REGISTER 0.8 us

0–3	4–7	8–11	12–15
6	5	Ra	BIT FIELD

$1 \rightarrow Ra_n$
$0's \rightarrow Ra_{0-(n-1)}$
$0's \rightarrow Ra_{(n+1)-15}$

A one is stored in register Ra in bit position n, where n is specified by the contents of the Bit Field. Zeros are stored in all other bit positions in register Ra.
Affected: Ra

LBRB LOAD BIT IN REGISTER AND BRANCH UNCONDITIONALLY 1.6 us

0–3	4–7	8–11	12–15
7	5	Ra	BIT FIELD
ADDRESS FIELD (0–15)			

$1 \rightarrow Ra_n$
$0's \rightarrow Ra_{0-(n-1)}$
$0's \rightarrow Ra_{(n+1)-15}$
$EA \rightarrow PR$

A one is stored in register Ra in bit position n, where n is specified by the contents of the Bit Field. Zeros are stored in all other bit positions in register Ra.

A branch is then executed unconditionally to the location specified by the contents of the second instruction word.

Affected: Ra

ABMM   ADD BIT IN MEMORY   3.47 us

0–3	4–7	8–11	12	13–15
8	0	BIT FIELD	I	Rxx
ADDRESS FIELD				

$(EA)+2^{15-n} \rightarrow EA$

The contents of the effective memory location are incremented by one in the bit position (n) designated in the Bit Field of the first instruction word. The result is stored in the effective memory location. Overflow will occur if the result is greater than $2^{15}-1$.

Affected: (Overflow), (EA)

ABMB   ADD BIT IN MEMORY AND BRANCH IF NONZERO   4.27 us

0–3	4–7	8–11	12	13–15
8	4	BIT FIELD	I	Rxx
OPERAND ADDRESS FIELD				
BRANCH ADDRESS FIELD				

0   15

$(EA)+2^{15-n} \rightarrow EA$

If Result $\neq 0$, $((PR)+2) \rightarrow PR$

If Result $=0$, $(PR)+3 \rightarrow PR$

The contents of the effective memory location are incremented by one in the bit position (n) designated in the Bit Field of the first instruction word. The result is stored in the effective memory location.

If the result is unequal to zero, a branch is executed to the location specified by the contents of the third instruction word. Only the direct address mode without indexing is performed. If the result equals zero, the next instruction in sequence is executed. Overflow will occur if the result is greater than $2^{15}-1$.

Affected: Overflow, (EA)

ABSM   ADD BIT IN MEMORY SHORT DISPLACED   2.67 us

0–3	4–7	8–11	12–15
9	0	BIT FIELD	DF

$((R1)+DF)+2^{15-n} \rightarrow (R1)+DF$

The contents of the effective memory location specified by the displacement field DF added to the contents of register R1 are incremented by one in the bit position (n) designated by the Bit Field. The result is stored in the effective memory location. Overflow will occur if the result is greater than $2^{15}-1$.

Affected: Overflow, (EA)

ABSB

ADD BIT IN MEMORY SHORT DISPLACED AND BRANCH IF NONZERO — 3.74 us

0 – 3	4 – 7	8 – 11	12 – 15
9	4	BIT FIELD	DF
BRANCH ADDRESS FIELD (0 – 15)			

$((R1)+DF)+2^{15-n} \rightarrow (R1)+DF$
If Result $\neq 0$, $((PR)+1) \rightarrow PR$
If Result $=0$, $(PR)+2 \rightarrow PR$

The contents of the effective memory location specified by the displacement field DF added to the contents of register R1 are incremented by one in the bit position (n) designated by the Bit Field. The result is stored in the effective memory location. Overflow will occur if the result is greater than $2^{15}-1$.

If the resulting word is unequal to zero, a branch is executed to the location specified by the second instruction word. The branch address may only be generated by the direct address mode without indexing. If the result equals zero, the next instruction in sequence is executed.

Affected: Overflow , (EA)

ABXM

ADD BIT IN MEMORY SHORT INDEXED — 2.67 us

0 – 3	4 – 7	8 – 11	12 – 15
9	8	BIT FIELD	Rx

$((Rx))+2^{15-n} \rightarrow (Rx)$

The contents of the effective memory location specified by the contents of register Rx are incremented by one in the bit position (n) designated by the Bit Field. The result is stored in the effective memory location. Overflow will occur if the result is greater than $2^{15}-1$.

Affected: Overflow , (EA)

ABXB

ADD BIT IN MEMORY SHORT INDEXED AND BRANCH IF NONZERO — 3.74 us

0 – 3	4 – 7	8 – 11	12 – 15
9	C	BIT FIELD	Rx
BRANCH ADDRESS FIELD (0 – 15)			

$((Rx))+2^{15-n} \rightarrow (Rx)$
If Result $\neq 0$ $((PR)+1) \rightarrow PR$
If Result $=0$ $(PR)+2 \rightarrow PR$

The contents of the effective memory location specified by the contents of register Rx are incremented by one in the bit position (n) designated by the Bit Field. The result is stored in the effective memory location. Overflow will occur if the result is greater than $2^{15}-1$.

If the resulting word is unequal to zero, a branch is executed to the location specified by the second instruction word. The branch address may only be generated by the direct address mode without indexing. If the result equals zero, the next instruction in sequence is executed.

Affected: Overflow, (EA)

ABR

ADD BIT IN REGISTER — 0.8 us

0–3	4–7	8–11	12–15
6	0	Ra	BIT FIELD

$(Ra)+2^{15-n} \rightarrow Ra$

The contents of register Ra are incremented by one in the bit position (n) designated in the Bit Field of the instruction word. The result is stored in register Ra. Overflow will occur if the result is greater than $2^{15}-1$.

Affected: Ra, Overflow

ABRB

ADD BIT IN REGISTER AND BRANCH IF NONZERO — 1.6 us

0–3	4–7	8–11	12–15
7	0	Ra	BIT FIELD
ADDRESS FIELD (0–15)			

$(Ra)+2^{15-n} \rightarrow Ra$
If Result $\neq 0$, EA $\rightarrow$ PR
If Result $=0$, $(PR)+2 \rightarrow PR$

The contents of register Ra are incremented by one in the bit position (n) designated in the Bit Field of the instruction word. The result is stored in register Ra. Overflow will occur if the result is greater than $2^{15}-1$.

If the result is unequal to zero, a branch is executed to the effective memory location. If the result equals zero, the next instruction in sequence is executed.

Affected: Ra Overflow

SBR

SUBTRACT BIT IN REGISTER — 0.8 us

0–3	4–7	8–11	12–15
6	1	Ra	BIT FIELD

$(Ra)-2^{15-n} \rightarrow Ra$

The contents of register Ra are decremented by one in the bit position (n) designated in the Bit Field of the instruction word. The result is stored in register Ra. Overflow will occur if the result is less than -2^{15}.

Affected: Ra, Overflow

SBRB

SUBTRACT BIT IN REGISTER AND BRANCH IF NONZERO 1.6 us

0 3	4 7	8 11	12 15
7	1	Ra	BIT FIELD
ADDRESS FIELD			

0 15

$(Ra)-2^{15-n} \rightarrow Ra$
If Result $\neq 0$, EA $\rightarrow$ PR
If Result $=0$, $(PR)+2 \rightarrow PR$

The contents of register Ra are decremented by one in the bit position (n) designated in the Bit Field of the instruction word. The result is stored in register Ra. Overflow will occur if the result is less than -2^{15}.

If the result is unequal to zero, a branch is executed to the effective memory location. If the result equals zero, the next instruction in sequence is executed.

Affected: Ra, (Overflow)

ZBMM

ZERO BIT IN MEMORY 3.47 us

0 3	4 7	8 11	12	13 15
8	1	BIT FIELD	I	Rxx
ADDRESS FIELD				

15

$0 \rightarrow (EA)_n$

The bit contained in the position in the effective memory location designated by the contents of the Bit Field (n) is cleared to zero. The other bits contained in the word are unaffected.
Affected: (EA)

ZBMB

ZERO BIT IN MEMORY AND BRANCH IF NONZERO 4.8 us

0 3	4 7	8 11	12	15
8	5	BIT FIELD	I	Rxx
OPERAND ADDRESS FIELD				
BRANCH ADDRESS FIELD				

15

$0 \rightarrow (EA)_n$
If Result $\neq 0$, $((PR)+2) \rightarrow PR$
If Result $=0$, $(PR)+3 \rightarrow PR$

The bit contained in the position in the effective memory location designated by the contents of the Bit Field (n) is cleared to zero. The other bits contained in the word are not affected. If the resulting word is unequal to zero, a branch is executed to the location specified by the contents of the third instruction word. Only the direct address mode without indexing is performed. If the result equals zero, the next instruction in sequence is executed.
Affected: (EA)

ZBSM

ZERO BIT IN MEMORY SHORT DISPLACED 2.67 us

0 - 3	4 - 7	8 - 11	12 - 15
9	1	BIT FIELD	DF

$0 \rightarrow ((R1)+DF)_n$

The bit n, designated by the Bit Field, contained in the memory location specified by the displacement field DF added to the contents of register R1 is cleared to zero. The other bits contained in the word are unaffected.
Affected: (EA)

ZBSB

ZERO BIT IN MEMORY SHORT DISPLACED AND BRANCH IF NONZERO 4.27 us

0 - 3	4 - 7	8 - 11	12 - 15
9	5	BIT FIELD	DF
BRANCH ADDRESS FIELD			

$0 \rightarrow ((R1)+DF)_n$
If Result $\neq 0$, $(PR)+1) \rightarrow PR$
If Result $=0$, $(PR)+2 \rightarrow PR$

The bit n, designated by the Bit Field, contained in the memory location specified by the displacement field DF added to the contents of register R1 is cleared to zero. The other bits contained in the word are unaffected. If the resulting word is unequal to zero, a branch is executed to the location specified by the second instruction word. The branch address may only be generated by the direct address mode without indexing. If the result equals zero, the next instruction in sequence is executed.
Affected: (EA)

ZBXM

ZERO BIT IN MEMORY SHORT INDEXED 2.93 us

0 - 3	4 - 7	8 - 11	12 - 15
9	9	BIT FIELD	Rx

$0 \rightarrow (Rx)_n$

The bit n, designated by the Bit Field, contained in the memory location specified by the contents of register Rx is cleared to zero. The other bits contained in the word are unaffected.
Affected: (EA)

ZBXB

ZERO BIT IN MEMORY SHORT INDEXED AND BRANCH IF NONZERO 4.27 us

0 - 3	4 - 7	8 - 11	12 - 15
9	D	BIT FIELD	Rx
BRANCH ADDRESS FIELD (0 - 15)			

$0 \rightarrow ((Rx))_n$
If Result $\neq 0$, $(PR)+1) \rightarrow PR$
If Result $=0$, $(PR)+2 \rightarrow PR$

The bit n, designated by the Bit Field, contained in the memory location specified by the contents of register Rx is cleared to zero. The other bits contained in the word are unaffected. If the resulting word is unequal to zero, a branch is executed to the location specified by the second instruction word. The branch address may only be generated by the direct address mode without indexing. If the result equals zero, the next instruction in sequence is executed.
Affected: (EA)

ZBR — ZERO BIT IN REGISTER — 0.8 us

0 – 3	4 – 7	8 – 11	12 – 15
6	2	Ra	BIT FIELD

$0 \rightarrow Ra_n$

The bit contained in the position in register Ra designated by the contents of the Bit Field (n) is cleared to zero. The other bits in register Ra are not affected.
Affected: Ra_n

ZBRB — ZERO BIT IN REGISTER AND BRANCH IF NONZERO — 1.6 us

0 – 3	4 – 7	8 – 11	12 – 15
7	2	Ra	BIT FIELD
ADDRESS FIELD (0 – 15)			

$0 \rightarrow Ra_n$
If Result $\neq 0$, EA $\rightarrow$ PR
If Result $=0$, (PR)+2 $\rightarrow$ PR

The bit contained in the position in register Ra designated by the contents of the Bit Field (n) is cleared to zero. The other bits in register Ra are not affected.

If the contents of Ra are not equal to zero, a branch is executed to the effective memory location. If the contents of Ra equals zero, the next instruction in sequence is executed.

Affected: Ra_n

OBMM — OR BIT IN MEMORY — 3.47 us

0 – 3	4 – 7	8 – 11	12	13 – 15
8	2	BIT FIELD	I	Rxx
ADDRESS FIELD (0 – 15)				

$1 \rightarrow (EA)_n$

The bit contained in the position in the effective memory location designated by the contents of the Bit Field (n) is set to one. The other bits contained in the word are unaffected.
Affected: (EA)

OBSM — OR BIT IN MEMORY SHORT DISPLACED — 2.67 us

0 – 3	4 – 7	8 – 11	12 – 15
9	2	BIT FIELD	DF

$1 \rightarrow ((R1)+DF)_n$

The bit n, designated by the Bit Field, contained in the memory location specified by the displacement field DF added to the contents of register R1 is set to one. The other bits contained in the word are unaffected.
Affected: (EA)

OBXM (5) OR BIT IN MEMORY SHORT INDEXED 2.94 us

0 3	4 7	8 11	12 15
9	A	BIT FIELD	Rx

$1 \rightarrow ((Rx))_n$

The bit n, designated by the Bit Field, contained in the memory location specified by the contents of register Rx is set to one. The other bits contained in the word are unaffected.
Affected: (EA)

OBR OR BIT IN REGISTER 0.8 us

0 3	4 7	8 11	12 15
6	3	Ra	BIT FIELD

$1 \rightarrow Ra_n$

The bit contained in the position in register Ra designated by the contents of the Bit Field (n) is set to one. The other bits in register Ra are not affected.
Affected: Ra_n

OBRB OR BIT IN REGISTER AND BRANCH UNCONDITIONALLY 1.6 us

0 3	4 7	8 11	12 15
7	3	Ra	BIT FIELD
ADDRESS FIELD (0–15)			

$1 \rightarrow Ra_n$
$EA \rightarrow PR$

The bit contained in the position in register Ra designated by the contents of the Bit Field (n) is set to one. The other bits in register Ra are not affected.

An unconditional branch is then executed to the effective memory location.

Affected: Ra_n

XBR EXCLUSIVE OR BIT IN REGISTER 0.8 us

0 3	4 7	8 11	12 15
6	4	Ra	BIT FIELD

$\overline{(Ra_n)} \rightarrow Ra_n$

The bit contained in the position in register Ra designated by the contents of the Bit Field (n) is complemented. The other bits in register Ra are not affected.
Affected: Ra_n

XBRB

EXCLUSIVE OR BIT IN REGISTER AND BRANCH IF NONZERO 1.6 us

0–3	4–7	8–11	12–15
7	4	Ra	BIT FIELD
ADDRESS FIELD (0–15)			

$(\overline{Ra_n}) \rightarrow Ra_n$
If Result $\neq 0$, EA $\rightarrow$ PR
If Result $= 0$, (PR)+2 $\rightarrow$ PR

The bit contained in the position in register Ra designated by the contents of the Bit Field (n) is complemented. The other bits in register Ra are not affected.

If the contents of Ra are unequal to zero, a branch is executed to the effective memory location. If the contents of Ra equal zero, the next instruction in sequence is executed.

Affected: Ra_n

TBMB

TEST BIT IN MEMORY AND BRANCH IF ONE 3.47 us

0–3	4–7	8–11	12	13–15
8	6	BIT FIELD	I	Rxx
OPERAND ADDRESS FIELD (0–15)				
BRANCH ADDRESS FIELD (0–15)				

If Effective Bit =1,
((PR)+2) $\rightarrow$ PR
If Effective Bit =0,
(PR)+3 $\rightarrow$ PR

The bit contained in the position in the effective memory location designated by the contents of the Bit Field(n)is tested. If the bit is equal to one, a branch is executed to the location specified by the contents of the third instruction word. Only the direct address mode without indexing is performed. If the tested bit is equal to zero, the next instruction in sequence is executed.
Affected: None

TBSB

TEST BIT IN MEMORY SHORT DISPLACED AND BRANCH IF ONE 3.2 us

0–3	4–7	8–11	12–15
9	6	BIT FIELD	DF
BRANCH ADDRESS FIELD (0–15)			

If Effective Bit =1,
((PR)+1) $\rightarrow$ PR
If Effective Bit =0,
(PR)+2 $\rightarrow$ PR

The bit n, designated by the Bit Field, contained in the memory location specified by the displacement field added to the contents of register R1 is tested. If the bit is equal to one, a branch is executed to the location specified by the contents of the second instruction word. Only the direct address mode without indexing is performed. If the tested bit is equal to zero, the next instruction in sequence is executed.
Affected: None

TBXB

TEST BIT IN MEMORY SHORT INDEXED AND BRANCH IF ONE — 3.2 us

0–3	4–7	8–11	12–15
9	E	BIT FIELD	Rx
BRANCH ADDRESS FIELD (0–15)			

If Effective Bit =1,
((PR)+1) → PR
If Effective Bit =0,
(PR)+2 → PR

The bit n, designated by the Bit Field, contained in the memory location specified by the contents of register Rx is tested. If the bit is equal to one, a branch is executed to the location specified by the contents of the second instruction word. Only the direct address mode without indexing is performed. If the tested bit is equal to zero, the next instruction in sequence is executed.

Affected: None

TBRB

TEST BIT IN REGISTER AND BRANCH IF ONE — 1.6 us

0–3	4–7	8–11	12–15
7	6	Ra	BIT FIELD
ADDRESS FIELD (0–15)			

If $(Ra_n)=1$, EA → PR
If $(Ra_n)=0$, (PR)+2 → PR

The bit contained in the position in Ra designated by the contents of the Bit Field (n) is tested. If the bit is equal to one, a branch is executed to the effective memory location. If the bit equals zero, the next instruction in sequence is executed.

Affected: None

CBMB

COMPARE BIT AND MEMORY — 4.27 us

0–3	4–7	8–11	12	13–15
8	7	BIT FIELD	I	Rxx
OPERAND ADDRESS FIELD				
BRANCH ADDRESS FIELD $(EA)=2^{15-n}$				
BRANCH ADDRESS FIELD $(EA)>2^{15-n}$				

If $2^{15-n}-(EA)=0$ ((PR)+2) → PR
If $2^{15-n}-(EA)<0$ ((PR)+3) → PR
If $2^{15-n}-(EA)>0$ (PR)+4 → PR

The contents of the effective memory location are algebraically subtracted from the value $+2^{15-n}$, where n is designated by the Bit Field in the first instruction word. If the result equals zero, a branch is executed to the location specified by the third instruction word. If the result is negative, a branch is executed to the location specified by the fourth instruction word. Only the direct addressing mode without indexing is permitted for the branch operation. If the result is greater than zero, the next instruction in sequence is executed. The contents of the memory location are not altered.

Affected: None

CBSB COMPARE BIT AND MEMORY SHORT DISPLACED

3.7 us

0 3	4 7	8 11	12 15
9	7	BIT FIELD	DF
BRANCH ADDRESS FIELD $(EA)=2^{15-n}$			
BRANCH ADDRESS FIELD $(EA)>2^{15-n}$			

0 15

If $2^{15-n} - (EA)=0$ $((PR)+2) \rightarrow PR$
If $2^{15-n} - (EA)<0$ $((PR)+3) \rightarrow PR$
If $2^{15-n} - (EA)>0$ $(PR)+4 \rightarrow PR$

The contents of the memory location specified by the displacement field DF added to the contents of register R1 are algebraically subtracted from the value 2^{15-n}, where n is designated by the Bit Field in the first instruction word. If the result equals zero, a branch is executed to the location specified by the second instruction word. If the result is negative, a branch is executed to the location specified by the third instruction word.

Only the direct addressing mode without indexing is permitted for the branch operation. If the result is greater than zero, the next instruction in sequence is executed. The contents of the memory location are not altered.
Affected: None

CBXB COMPARE BIT AND MEMORY SHORT INDEXED

3.7 us

0 3	4 7	8 11	12 15
9	F	BIT FIELD	Rx
BRANCH ADDRESS FIELD $(EA)=2^{15-n}$			
BRANCH ADDRESS FIELD $(EA)>2^{15-n}$			

0 15

If $2^{15-n} - (EA)=0$ $((PR)+2) \rightarrow PR$
If $2^{15-n} - (EA)<0$ $((PR)+3) \rightarrow PR$
If $2^{15-n} - (EA)>0$ $(PR)+4 \rightarrow PR$

The contents of the memory location specified by the contents of register Rx are algebraically subtracted from the value 2^{15-n}, where n is designated by the Bit Field in the first instruction word. If the result equals zero, a branch is executed to the location specified by the second instruction word. If the result is negative, a branch is executed to the location specified by the third instruction word.
Only the direct addressing mode without indexing is permitted for the branch operation. If the result is greater than zero, the next instruction in sequence is executed. The contents of the memory location are not altered.
Affected: None

GMR   GENERATE MASK IN REGISTER   0.8 us

0 – 3	4 – 7	8 – 11	12 – 15
6	7	Ra	BIT FIELD

1's → Ra_{0-n}
0's → Ra_{n+1-15}

Ones are stored in register Ra in bit position Ra_0 through Ra_n, where n is specified by the contents of the bit field. Zeroes are stored in register Ra in bit positions Ra_{n+1} through Ra_{15}. Overflow will result and PR_0 will be set if n = 0.

Affected: Ra, (Overflow)

GMRB   GENERATE MASK IN REGISTER AND BRANCH UNCONDITIONALLY   1.6 us

0 – 3	4 – 7	8 – 11	12 – 15
7	7	Ra	BIT FIELD
ADDRESS FIELD (0 – 15)			

1's → Ra_{0-n}
0's → Ra_{n+1-15}
EA → PR

Ones are stored in register Ra in bit positions Ra_0 through Ra_n, where n is specified by the contents of the bit field. Zeroes are stored in register Ra in bit positions Ra_{n+1} through Ra_{15}. Overflow will result and be set if n=0.

A branch is then executed unconditionally to the location specified by the contents of the second instruction word.

Affected: Ra, Overflow

BYTE MANIPULATION INSTRUCTIONS

These instructions enable bytes to be moved and interchanged in the general register file. All of these instructions contain two register addresses Ra and Rb. By making Ra equal to Rb, either byte or both bytes can be moved within one register. By making Ra unequal to Rb bytes can be moved from register to register. The move instructions cause one byte to be cleared to zero in the destination register, whether Ra=Rb or Ra≠Rb.

MUR

MOVE UPPER BYTE REGISTER TO REGISTER 0.8 us

0 3	4 7	8 11	12 15
0	B	Ra	Rb

$(Rb_{0-7}) \rightarrow Ra_{0-7}$
$0 \rightarrow Ra_{8-15}$

The more significant byte stored in register Rb is transferred to the more significant byte position of register Ra. Zeros are transferred to the less significant byte position in register Ra. If Ra=Rb, the instruction becomes a "Clear Lower Byte in Register" instruction.
Affected: Ra

MLR

MOVE LOWER BYTE REGISTER TO REGISTER 0.8 us

0 3	4 7	8 11	12 15
0	C	Ra	Rb

$(Rb_{8-15}) \rightarrow Ra_{8-15}$
$0 \rightarrow Ra_{0-7}$

The less significant byte stored in register Rb is transferred to the less significant byte position of register Ra. Zeros are transferred to the more significant byte position in register Ra. If Ra=Rb, this instruction becomes a "Clear Upper Byte in Register" instruction.
Affected: Ra

MBR

MOVE BYTE RIGHT REGISTER TO REGISTER 0.8 us

0 3	4 7	8 11	12 15
0	8	Ra	Rb

$(Rb_{0-7}) \rightarrow Ra_{8-15}$
$0 \rightarrow Ra_{0-7}$

The more significant byte stored in register Rb is transferred to the less significant byte position of register Ra. Zeros are transferred to the more significant byte position in register Ra. If Ra=Rb, this instruction becomees a fast "Logical Right Shift Eight Bits" instruction.
Affected: Ra

MBL MOVE BYTE LEFT REGISTER TO REGISTER 0.8 us

0–3	4–7	8–11	12–15
0	9	Ra	Rb

$(Rb_{8-15}) \rightarrow Ra_{0-7}$
$0 \rightarrow Ra_{8-15}$

The less significant byte stored in register Rb is transferred to the more significant byte position of register Ra. Zeros are transferred to the less significant byte position in register Ra. If Ra=Rb, this instruction becomes a fast "Logical Left Shift Eight Bits" instruction.
Affected: Ra

IBR INTERCHANGE BYTES REGISTER TO REGISTER 0.8 us

0–3	4–7	8–11	12–15
0	A	Ra	Rb

$(Rb_{0-7}) \rightarrow Ra_{8-15}$
$(Rb_{8-15}) \rightarrow Ra_{0-7}$

The less significant byte stored in register Rb is transferred to the more significant byte position in register Ra, and the more significant byte stored in register Rb is transferred to the less significant byte position in register Ra. If Ra = Rb, this instruction becomes a "Rotate Eight Bits" instruction.
Affected: Ra

UNCONDITIONAL BRANCH INSTRUCTIONS

This group includes the Branch and the Branch and Link instructions. All are unconditional branch instructions. Each time a branch is executed all 16 bits of the Program Register, are replaced.

BLM BRANCH AND LINK 1.6 us

0 3	4 7	8 11	12	13 15
E	7	Ra	I	Rxx
ADDRESS FIELD (0 15)				

(PR)+2 → Ra
EA → PR

The 16-bit contents of the Program Register replace the contents of register Ra and then the effective memory address replaces the contents of the Program Register.
Affected: Ra

BLI BRANCH AND LINK IMMEDIATE 0.8 us

0 3	4 7	8 11	12 15
E	F	Ra	0

(PR)+2 → Ra
(PR)+1 → PR

The 16-bit contents of the Program Register replace the contents of register Ra and then the next instruction in sequence is executed.
Affected: Ra

BRU BRANCH UNCONDITIONALLY 1.6 us

0 3	4 7	8 11	12	13 15
E	7	0000	I	Rxx
ADDRESS FIELD (0 15)				

EA → PR

The 16 bits of the effective memory address replace the contents of the Program Register.
Affected: None

HOP BRANCH SHORT DISPLACED 1.07 us

0 – 3	4 – 7	8 – 11	12 – 15
F	7	0	DF

0 15

(PR)+DF → PR

The contents of the displacement field DF are added to the contents of the Program Register. The result is then stored in the Program Register. Therefore a branch is executed which has a range of from zero to +15 locations with respect to the current program location.

Affected: None

BRX BRANCH SHORT INDEXED 1.07 us

0 – 3	4 – 7	8 – 11	12 – 15
F	F	0	Rx

(Rx) → PR

The 16 bit contents of register Rx replace the contents of the Program Register.

Affected: None

CONTROL INSTRUCTIONS

This group includes Halt, No Operation, Set Protect Register, Set Lower Protect Register and set Upper Protect Register.

HLT

HALT

0 – 3	4 – 7	8 – 11	12 – 15
0	0	//////////	//////////

Program Execution is halted until the program is manually restarted. The halt occurs after the Program Register is advanced and the next instruction is transferred to the instruction register.

Affected: None

NOP

NO OPERATION

0.8 us

0 – 3	4 – 7	8 – 11	12 – 15
6	6	//////////	//////////

The execution of this instruction does not modify the contents of any general register or memory location.

Affected: None

SPR

SET PROTECT REGISTER

0.8 us

0 – 3	4 – 7	8	9 10 11	12 13 14 15
0	2	0	X X P0	P1 P2 P3 P4

P0 → B3R0
P1 → B3R1
P2 → B3R2
P3 → B3R3
P4 → B3R4
1 → B3R5-B3R7

The five least significant bits of the Instruction Word (IR11-IR15) are transferred directly to the Protect Boundary Register #3 (Bits 0 thru 4). Ones are loaded into bits five thru seven of this register therby yielding a 2K word protect granularity.

Affected: Protect Boundary Register #3.

SGP

SET GLOBAL PROTECT REGISTER

0.8 us

0–3	4–7	8	9	10	11	12–15
0	2	1	X	X	X	Rb

$(Rb)_0$ B3R0
$(Rb)_1$ B3R1
$(Rb)_2$ B3R2
$(Rb)_3$ B3R3
$(Rb)_4$ B3R4
$(Rb)_5$ B3R5
$(Rb)_6$ B3R6
$(Rb)_7$ B3R7

The eight most significant bits of the GPR specified by the Rb field are transferred directly to Protect Boundary Register #3 (Bits 0 thru 7) thereby yielding a 256 word protect granularity.

Affected: Protect Boundary Register #3.

SLP

SET LOWER PROTECT REGISTER

0.8 us

0–3	4–7	8–11	12–15
0	3	//////	Rb

$(Rb)_0 \rightarrow$ LPR0
⋮ ⋮ ⋮
$(Rb)_8 \rightarrow$ LPR8

The nine most significant bits of the GPR specified by the Rb field are transferred directly to Lower Protect Boundary Register, (Bits 0 thru 8) thereby yielding a 128 word protect granularity.

Affected: Lower Protect Boundary Register.

SUP

SET UPPER PROTECT REGISTER

0.8 us

0–3	4–7	8–11	12–15
0	4	//////	Rb

$(Rb)_0 \rightarrow$ UPR0
⋮ ⋮ ⋮
$(Rb)_8 \rightarrow$ UPR8

The nine most significant bits of the GPR specified by the Rb field are transferred directly to the Upper Protect Boundary Register, (Bits 0 thru 8) thereby yielding a 128 word protect granularity.

Affected: Upper Protect Boundary Register.

INTERRUPT AND CALL INSTRUCTIONS

The interrupt instructions provide the capability for complete program manipulation of the states of all three latches present in each priority interrupt level. The Request Executive Service is the executive call instruction and the Request Multiprocessor Interrupt instruction enables each CPU in a multiprocessor configuration to produce an interrupt in the other cpu.

Each interrupt instruction contains a binary coded level field. This field permits each of the 16 (maximum) levels to be addressed and operated on individually.

SIE

SET INTERRUPT ENABLE 1.33 us

0 3	4 7	8 11	12 15
2	6	0 1 0 0	LEVEL

$1 \rightarrow ENA_G$, Level

Enable the priority interrupt level specified by the Level selection field.

The PFS/AS, Memory Parity, Unimplemented Instruction, System Protect and Floating Point Overflow Trap interrupt levels are always enabled when present in the system. The Enable latch state of these levels cannot be altered by instruction execution.

Affected: None

RIE

RESET INTERRUPT ENABLE 1.33 us

0 3	4 7	8 11	12 15
[illegible]	7	0 1 0 0	LEVEL

$0 \rightarrow ENA_G$, Level

Disable the priority interrupt level specified by the Level selection field.

The PFS/AS, Memory Parity, Unimplemented Instruction, System Protect and Floating Point Overflow Trap interrupt levels are always enabled when present in the system. The Enable latch state of these levels cannot be altered by instruction execution.

SIR SET INTERRUPT REQUEST 1.33 us

0 3	4 7	8 10	11 15
2	6	1 0 0	LEVEL

$1 \rightarrow REQ_{G,\ Level}$

Set the Request latch of the priority interrupt level specified by the Level selection field. NOTE: Levels C_{16} and D_{16} should not be requested by the program.

Affected: None

RIR RESET INTERRUPT REQUEST 1.33 us

0 3	4 7	8 10	11 15
2	7	1 0 0	LEVEL

$0 \rightarrow REQ_{G,\ Level}$

Reset the Request latch of the priority interrupt level specified by the Level selection field.

Affected: None

SIA SET INTERRUPT ACTIVE 1.33 us

0 3	4 7	8 10	11 15
2	6	0 0 0	LEVEL

$1 \rightarrow ACT_{G,\ Level}$

Activate the priority interrupt level specified by the Level selection field.
NOTE: Level 0 (Power Fail/Auto Start) may not be set active by the program.

Affected: None

RIA RESET INTERRUPT ACTIVE 1.33 us

0 3	4 7	8 10	11 15
2	7	0 0 0	LEVEL

$0 \rightarrow ACT_{G,\ Level}$

Deactivate the priority interrupt level specified by the Level selection field.

Affected: None

REX REQUEST EXECUTIVE SERVICE 1.07 us

0 3	4 7	8 15
2	3	Service Field

$1 \rightarrow REQ_{UI}$

An interrupt request signal is sent to the Unimplemented Instruction Trap level. The executive service requested is defined by the contents of the Service Field. The program count is not advanced before the trap is generated. Therefore the stored PSW contains the address of the REX service.

The Unimplemented Instruction Trap level will become active and the interrupt routine entered at the completion of execution of the REX instruction, provided that neither this level nor any higher level is already active. If this level or a higher level is active, execution of the REX cannot be completed. The machine must be manually cleared and restarted if this error condition occurs.

Affected: None

RMI

REQUEST MULTIPROCESSOR INTERRUPT 0.8 us

0–3	4–7	8–15
0	1	//////////

1 REQ_X in other CPU

A pulse is generated by the executing CPU which requests the interprocessor communication interrupt in the other CPU.

Affected: None

CAR

CLEAR ACTIVE AND RETURN 1.87 us

0–3	4–7	8–11	12–15
2	4	0	0

0 → $ACT_{Highest\ Active}$

The Active latch of the highest active interrupt level is cleared and the 16 bit contents of the memory location dedicated to the highest active level ~~and~~ transferred to the Program Register. If no interrupt is active when the CAR instruction is executed, the contents of location 0 are transferred to the Program Register. If an Interrupt is requesting when the CAR instruction is executed, (the interrupt) will not go in service until 1 instruction after the CAR is executed.

Affected: None

CIR

CLEAR INTERRUPT AND RETURN 1.87 us

0–3	4–7	8–11	12–15
2	5	0	0

0 → $ACT_{Highest\ Active}$

0 → $REQ_{Highest\ Active}$

Both the Active and Request latches of the highest active interrupt level are cleared and the 16 bit contents of the memory location dedicated to the highest active level are transferred to the Program Register. If no interrupt is active when the CIR instruction is executed, the contents of location 0 are transferred to the Program Register.

Affected: None

INPUT/OUTPUT INSTRUCTIONS

Two input instructions are provided to enable a data or status word to be transferred from any peripheral device to any general register. Two output instructions are provided to enable a data or command word to be transferred from any general register to any peripheral device. Some peripheral devices such as the disc transfer data only under control of the Direct Memory Processor. Therefore, only command and status words are transferred under program control to/from these devices.

Up to 64 peripheral devices, consisting of four groups of 16 each, are addressable by each instruction. The group address is obtained from the two least significant bits of the operation code field. Therefore four operation codes and mnemonics are assigned to each instruction.

I/O GROUP A	Consists of device addresses 00-0F
I/O GROUP B	Consists of device addresses 10-1F
I/O GROUP C	Consists of device addresses 20-2F
I/O GROUP D	Consists of device addresses 30-3F

All instructions are executed in the fixed length of time contained in each instruction description.

ISA	ISA	(48)	Input Status From I/O Group A
ISB	ISB	(49)	Input Status From I/O Group B
ISC	ISC	(4A)	Input Status From I/O Group C
ISD	ISD	(4B)	Input Status From I/O Group D

1.6 us

0–3	4–5	6–7	8–11	12–15
4	10_2	G	Ra	D

G, D → I/O Address Lines
Device Status → Ra

The group (G) and device (D) numbers contained in the instruction word are placed on the I/O bus address lines.

Up to 16 bits of status are then transferred from the addressed device over the I/O bus to replace the contents of register Ra.
Affected: Ra

IDA
IDB
IDC
IDD

IDA (4C) Input Data From I/O Group A

IDB (4D) Input Data From I/O Group B

IDC (4E) Input Data From I/O Group C

IDD (4F) Input Data From I/O Group D

1.6 us

0 – 3	4 – 5	6 – 7	8 – 11	12 – 15
4	11_2	G	Ra	D

G, D → I/O Address Lines

Device Data → Ra

The group (G) and device (D) numbers contained in the instruction word are placed on the I/O bus address lines.

Up to 16 bits of data are then transferred from the addressed device over the I/O bus to replace the contents of register Ra.

Affected: Ra

OCA
OCB
OCC
OCD

OCA (40) Output Command To I/O Group A

OCB (41) Output Command To I/O Group B

OCC (42) Output Command To I/O Group C

OCD (43) Output Command To I/O Group D

1.33 us

0 – 3	4 – 5	6 – 7	8 – 11	12 – 15
4	00_2	G	Ra	D

G, D → I/O Address Lines

(R_a) → I/O Data Lines

The group (G) and device (D) numbers contained in the instruction word are placed on the I/O bus address lines.

The 16 bit output command stored in register Ra is then transferred to the I/O register and placed on the I/O bus data lines.

ODA
ODB
ODC
ODD

ODA (44) Output Data To I/O Group A

ODB (45) Output Data To I/O Group B

ODC (46) Output Data To I/O Group C

ODD (47) Output Data To I/O Group D

1.33 us

0 3	4 5	6 7	8 11	12 15
4	01_2	G	Ra	D

G, D → I/O Address Lines

(R_a) → I/O Data Lines

The group (G) and device (D) numbers contained in the instruction word are placed on the I/O bus address lines.

The 16 bit data word stored in register Ra is then transferred to the I/O register and placed on the I/O bus data lines.

Three signals are needed to enable command decode. They are:

DRIOFN - Input/Output Function
DRCDFN - Command/Data Function
DRIOSN - I/O Sync

Data flow is determined by DRIOFN. If DRIOFN is true (low), data is input. If DRIOFN is false (high), data is output.

DFCDFN determines whether the instruction is a command or data. If DFCDFN is true (low), the instruction is data. If DRCDFN is false (high), the instruction is a command (or status).

These lines are interrogated at I/O sync time, DRIOSN.

	DRCDFN	$\overline{\text{DRCDFN}}$
DRIOFN	Input Data IDA	Input Status ISA
$\overline{\text{DRIOFN}}$	Output Data ODA	Output Command OCA

IV. PRIORITY INTERRUPTS

OVERVIEW

The MODCOMP II priority interrupt system contains three standard levels expandable in increments of four levels up to a total of 16 levels including the optional Power Fail Safe/Auto Start. Each external level can be selectively enabled and disabled under program control. Internal levels 0, 1, 2, 4, and 5, when present, are always enabled. In addition, the recognition of interrupt signals can be deferred for all interrupt levels below a selected level. Furthermore, interrupt request signals can be generated by instruction execution.

Of the three standard interrupt levels, two are I/O interrupt levels which have party line interrupt structures with 17 sub-priorities each and automatic source identification for up to 64 devices.

Each priority level is assigned two dedicated memory locations for the entry and return addresses unique to that level. The entry address of the interrupt processing routine is stored in one dedicated location. The return address, which is the contents of the Program Register (PR), is stored in the other dedicated location at the time the interrupt routine was entered. The 64 sub-levels of each I/O interrupt level share the return address of that level but are assigned unique entry address locations.

Nested interrupt routine execution is automatically handled for the 16 priority levels. The sub-levels of each I/O priority interrupt level cannot interrupt each other, but if several attempt to interrupt at the same time, the highest priority sub-level is recognized first.

LEVEL ASSIGNMENTS

The dedicated memory locations for each interrupt level and the signals connected to these levels are shown in Table 4-1.

There are three standard interrupt levels (4,C, and D) present in each MODCOMP II and they are connected to Unimplemented Instruction Trap and to the I/O Data and Service party lines. Power Fail Safe/Auto Start is standard in some models and optional in others. It is connected to the highest priority level (0).

The first optional group of interrupts (5,6,E, and F) are dedicated to the Executive Features Option. The second optional group of interrupts (1,2,3, and 7) are assigned to the System Protect option with level 7 available for an external interrupt signal. Priority level F is dedicated to the Task Scheduler Interrupt which allows the MAX III Executive to maintain a software task priority queue below the hardware priority queue. Levels 8,9,A,B are available as an optional group for connection to external equipment.

MODEL NUMBER	MEMORY LOCATION$_{16}$	LEVEL$_{16}$	PROGRAM LINKAGE	INTERRUPT SIGNAL
3739	20	0	Return	Power Fail Safe/Auto Start
	21		Entry	
3731	22	1	Return	Memory Parity
	23		Entry	
3731	24	2	Return	System Protect
	25		Entry	
3731	26	3	Return	Multiprocessor Communications
	27		Entry	
STD.	28	4	Return	Unimplemented Instruction Trap
	29		Entry	
3730	2A	5	Return	Floating Point Overflow
	2B		Entry	
3730	2C	6	Return	Real Time Clock
	2D		Entry	
3731	2E	7	Return	External
	2F		Entry	
3732	30	8	Return	External
	31		Entry	
3732	32	9	Return	External
	33		Entry	
3732	34	A	Return	External
	35		Entry	
3732	36	B	Return	External
	37		Entry	
STD.	38	C	Return	I/O Data Party Line
	39		Not Used	
STD.	3A	D	Return	I/O Service Party Line
	3B		Not Used	
3730	3C	E	Return	Console Interrupt
	3D		Entry	
3730	3E	F	Return	Task Scheduler
	3F		Entry	

TABLE 4-1 INTERRUPT LEVEL ASSIGNMENTS

INTERRUPT OPERATION AND PROGRAM CONTROL

Each interrupt level contains three flip-flops which collectively define the state of the level.

The Request flip-flop is set by the external interrupt request signal or by execution of the Set Interrupt Request (SIR) instruction. The purpose of this flip-flop is to store the request until it can be processed by the computer. It is reset by execution of either the Clear Interrupt and Return (CIR) or the Reset Interrupt Request (RIR) instruction.

The Enable flip-flop, when set, permits the stored request to interrupt the program. This flip-flop is set by execution of the Set Interrupt Enable (SIE) instruction and reset by execution of the Reset Interrupt Enable (RIE) instruction.

The Active flip-flop is set when the program interrupt signal is generated. It is not reset, except by execution of the Reset Interrupt Active (RIA) instruction, until the Clear Interrupt and Return (CIR) instruction is executed to exit an interrupt routine. Therefore, it indicates that an interrupt was being processed at this level and enables program control to be returned to the level if one or more higher priority interrupts occurred while the level was being serviced. The Clear Active and Return (CAR) operates just as the Clear Interrupt and Return (CIR) except that the request latch is not reset, thus allowing new reponses to be acknowledged that may have occurred while the level was active. The Set Interrupt Active (SIA) and Reset Interrupt Active (RIA) instructions are provided to enable a level to be made active without causing a program interruption. Program interruption can be deferred from the level made active down through all lower levels by execution of these two instructions.

Operation of the Master Clear switch resets all three flip-flops in each level, except the Enable flip-flops for levels 0, 1, 2, 4, and 5 (Power Fail-Safe/Auto Start, Memory Parity, System Protect, Unimplemented Instruction Trap and Floating Point Overflow).

Based on the operation of the three interrupt level flip-flops, the conditions necessary for interrupting the computer from a given interrupt level are:

- The level must be enabled
- A request signal must have occurred
- No higher priority level must be active
- The execution of the current instruction must be completed

When these conditions are met, program switching occurs. The current 16-bit contents of the Program Register are stored in the Return location assigned to the interrupting level. The 16-bit contents of the Entry location assigned to the level are then transferred to the Program Register, and the execution of the interrupt routine is started. Program switching requires 2.4 μsec.

The interrupt level is cleared by execution of the Clear Interrupt and Return instruction. Execution of this instruction clears the Request and Active flip-flops of the highest active level and transfers the 16-bit contents of the dedicated return location to the Program Register.

INTERRUPT SUB-LEVEL OPERATION AND PROGRAM CONTROL

The I/O data and Service Interrupt levels, C and D, each provide 17 sub-priorities which are assigned to peripheral devices and to external equipment (refer to Table 4-2). Addresses and dedicated memory locations are available for a total of 64 sub-levels for each of the two interrupt levels. The higher transfer rate devices such as the discs and analog input subsystems are assigned to the higher priority sub-levels. The data and service interrupt priorities are identical for each peripheral device. Each data and service sub-level is identified by the dedicated memory locations for storing subroutine entry addresses. Sub-levels of a given priority interrupt level cannot interrupt each other, but if several attempt to interrupt at the same time, the highest priority sub-level is recognized first. Any data interrupt sub-level can interrupt any service interrupt sub-level so that data transfers have precedence over error or status checking routines.

I/O PRIORITY SUB-LEVEL	INTERRUPT LOCATION		PERIPHERAL DEVICE
	DATA	SERVICE	
0	81	C1	Moving Head Disc
1	82	C2	Fixed Head Disc
2	90,91	D1	High Level Analog Input Subsystems
3	98,99	D8,D9	Communications Multiplexer
4	83	C3	High Performance Magnetic Tape
5	92,93	D3	Wide Range Analog Input Subsystem (#1)
6	A0-AF	E0-E7	Input/Output Interface Subsystem (#1)
6	A0-AB	E0-EB	MODAC Subsystem (#1)
7	84	C4	Moderate Performance Magnetic Tape
8	85	C5	Card Readers (300-1000 CPM)
9	86	C6	Card Punch
10	94	D4	Wide Range Relay Analog Input Subsystem
11	87	C7	Line Printer (600 LPM)
12	88	C8	X-Y Plotter
12	88	C8	Electrostatic Printer/Plotter
13	89	C9	High Speed Paper Tape Punch
14	8B	CB	Line Printer 50-150 LPM
15	8A	CA	Teletype/Paper Tape Reader
16	B0-BF	F0-FF	Input/Output Interface Subsystem (#2)
16	B0-BB	F0-FB	MODAC Subsystem (#2)

TABLE 4-2 Sub-Level Assignments

When a Data Interrupt is serviced (level C_{16}), the contents of the Program Register are stored in memory location 38 and are replaced by the contents of the data interrupt entry location (80 to BF) for the highest priority peripheral device. The contents of the entry location point to the unique interrupt subroutine for that I/O device.

The interrupt subroutine is exited with a CIR instruction which clears the interrupt level and branches to the address contained in memory location 38. If another Data Interrupt request is pending, the interrupt processing routine is re-entered immediately.

When a Service Interrupt is serviced, the operation is identical except that the entry and return addresses are 3A and 3B with dedicated sub-level interrupt locations C0-FF.

For additional information on the programming of the I/O interrupts, refer to Section V.

TRAPS

Traps are defined as conditions which cause the execution of the current instruction to be aborted before completion and generate an interrupt request signal. Only these internal conditions operate as traps:

- Unimplemented Instruction
- Memory Parity
- System Protect Violation
- Floating Point Overflow

Unimplemented Instruction Trap

An Unimplemented Instruction Trap occurs upon the execution of a REX instruction or when macro instructions (Op Codes $1X_{16}$, $3X_{16}$, or $5X_{16}$) are attempted. The floating point set belongs to the macro group and therefore floating point instructions are trapped when this option is not present in a computer.

When this trap occurs, the contents of the Program Register point to the memory location which contains the unimplemented instruction. Therefore, the instruction can be examined and be simulated by a subroutine. Keeping the contents of the Program Register from being advanced until the Unimplemented Instruction interrupt (level 4) becomes active, means that unimplemented instructions must not be present in any higher level interrupt routines.

The Unimplemented Instruction interrupt level is always enabled. It cannot be disabled by instruction execution. This condition prevents the possibility of stalling the machine due to an unimplemented instruction occurring when the level is disabled.

The Opcodes which have no assigned mnemonic are undefined instructions and their operation is unspecified. These undefined instructions will not generate an unimplemented instruction trap nor will they be executed as NOPs. These Opcodes should not be used.

Memory Parity Trap

Instructions which result in a memory parity trap are aborted. If the optional memory parity interrupt level is not present, the computer will suspend all operations until the master clear switch is depressed. If the memory parity interrupt level is present, the interrupt will be processed in the normal fashion. If a parity error occurs during a higher priority interrupt subroutine (Power Fail Safe/ Auto Start) the interrupt will not occur until the higher level is cleared. The memory parity interrupt is always enabled.

System Protect

This trap is used by the MAX III Modular Applications Executive to allow the checkout and execution of programs in unprotected areas of memory without interfering with the execution or integrity of programs residing in the protected areas of memory. This trap occurs if a memory protect violation, privileged instruction violation, or illegal branch is attempted by the unprotected program. The trap mechanism returns control to MAX III and results in the immediate aborting of the offending program.

A memory protect violation occurs when an attempt is made to write into protected memory by an instruction contained in unprotected memory. At this time a trap will occur and prevent the illegal write operation from occurring. If a branch is attempted into protected memory either directly or indirectly (a short indexed operation, for example), by an instruction or indirect address contained in unprotected memory, the branch occurs before the trap is implemented. The PR will be updated by the branch and will contain the address of protected memory into which the branch was made.

A privileged instruction violation occurs when a program in unprotect memory attempts to execute any CONTROL instruction, INPUT/OUTPUT instruction or INTERRUPT AND CALL instruction except REX.

The System Protect feature is enabled and disabled by the console key switch.

Floating Point Overflow

Floating point operands presented to the floating point unit must be normalized. However, floating point overflow or underflow will occur if the resultant exponent of a floating point operation is unable to be expressed within the range of the nine bit binary exponent field of the floating point format.

If the resultant floating point fraction must be left shifted to normalize, the binary exponent must be decremented by one for each bit position left shifted. If the exponenet decrements past all zeroes to all ones, floating point underflow has occurred.

If the resultant floating point fraction must be right shifted to be normalized, the binary exponent must be incremented by one for each bit position right shifted. If the exponent increments past all ones to all zeroes, floating point overflow has occurred.

Either occurrence causes the floating point overflow trap mechanism to terminate the normal FPU operation and does not allow any results to be transferred back to the CPU register file. The original register operands are maintained in the CPU register file and may be interrogated for further overflow/underflow clarification.

The floating point overflow trap level, when present in the system, is Level 5 and is always enabled.

POWER FAIL SAFE/AUTO START INTERRUPT

When the a-c line voltage drops below 105 volts, an interrupt is generated a minimum of 200 memory cycles before the memory write current is disabled. This feature allows the inclusion and execution of a user supplied power failure interrupt routine to store all operands and I/O status, for example, which protects the integrity of the operating program stored in memory during transient or long term power failure conditions.

Upon the generation of a power failure interrupt, the Program Register is stored in memory location 20 and the power failure routine is entered using the address stored in location 21.

When a-c power is restored, the system is normalized, an interrupt is generated, and the start-up routine is entered using the address stored in location 21. The initial address of the auto-start subroutine should be stored in location 21 by the power failure subroutine. This level is always enabled.

V. INPUT OUTPUT

OVERVIEW

The basic I/O facility of the MODCOMP II computer consists of a time-shared (party line) I/O bus capable of transferring data, commands and device status. Data can be transferred between any general register and any of up to 64 addressable peripheral devices. Up to 16 bits can be transferred in parallel over the bus under program control. In addition, the Direct Memory Processor (DMP) is available as an optional I/O facility which permits transfer of blocks of data to and from memory on a cycle stealing basis.

Figure 5-1 is the input/output subsystem block diagram. The I/O bus and a typical peripheral device controller are shown in addition to the computer I/O subsystem.

INSTRUCTION EXECUTION SEQUENCE

The execution sequence for all I/O instructions - Input Data, Input Status, Output Data, Output Command - consists of:

(1) The device address consisting of bits 6, 7 and 12-15 are transferred from the instruction register, through the I/O Control (Figure 5-1) to the addressed peripheral device controller.

(2) A set of control signals are sent to the addressed controller which define the operation - Input Data, Input Status, Output Data, or Output Command.

(3) If the control signals call for an input, the device places a data word or status word on the 16 data lines of the I/O bus and this word is then transferred to register Ra as specified by the instruction. The fixed execution time for all input instructions is 1.6 microseconds. If the control signals call for an output, the contents of register Ra, as defined in the instruction word, are transferred to the output buffer register and then placed on the 16 data lines of the I/O bus. Execution of all output instructions is completed in a total of 1.33 microseconds.

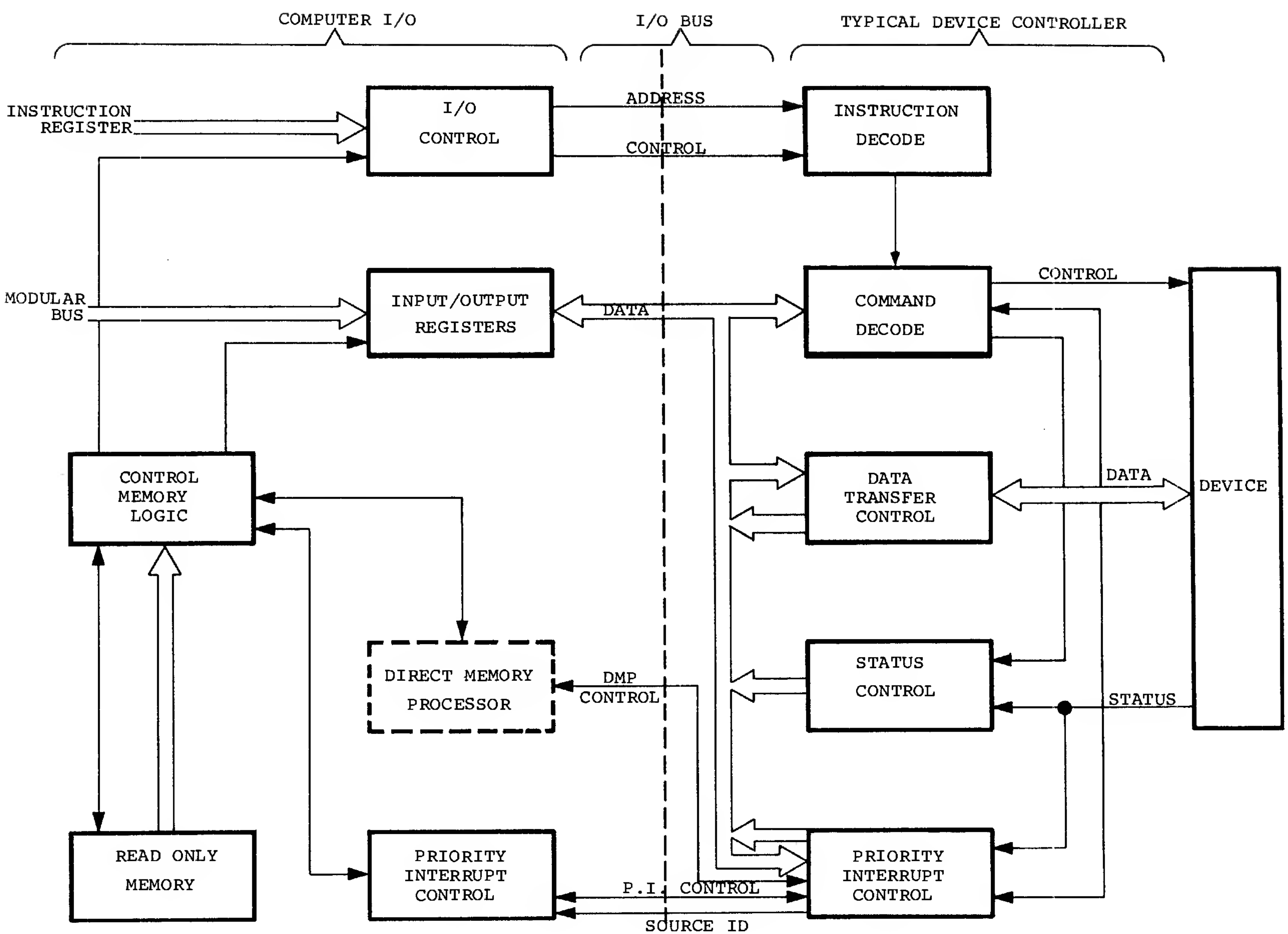
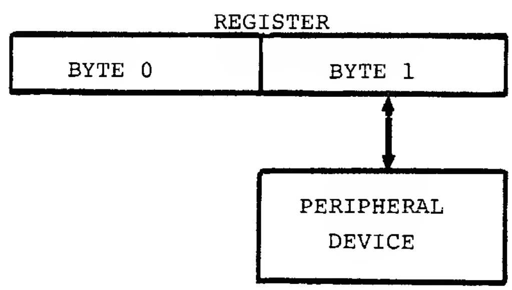


Figure 5-1 Input/Output Subsystem Block Diagram

TRANSFER FORMATS

Data is transferred over the I/O Bus as a 16-bit word. If a peripheral device requires or generates a data word of less than 16 bits, the device data word occupies the least significant bits of the 16-bit CPU data word with the unused bits appearing as zeroes. When less than 16 bits are output to a device, the outputs are taken from the less significant end of the register Ra or memory and zeroes are stored in the otherwise unused bits at the most significant end. This format is consistent with the operation of the byte manipulation instructions.

BYTE TRANSFER FORMAT



REGISTER I/O TRANSFER MODES

Three transfer modes are available for program controlled transfers.

The interrupt mode can be used with any device which generates a transfer request signal. This group of devices includes all standard computer peripherals. The transfer request signal is connected to an interrupt level. Interrupt service routines can perform transfers and all required overhead functions at rates up to approximately 60K words per second.

The test and transfer mode is performed by first testing a device by means of the Input Status instruction. When the "Data Ready" status bit equals zero, a transfer can be made to or from the addressed device. The maximum transfer rate in this mode is determined almost entirely by the timing of the device.

The burst mode can be used with devices which can perform a word transfer any time the computer executes an I/O instruction addressed to the device. Output bursts of up to 15 words (one per register) can be performed at the burst rate of 750K words per second. Input bursts can be performed at 625K words per second. This mode is useful in applications such as updating a group of digital-to-analog converter registers.

INPUT/OUTPUT INTERRUPTS

Two standard I/O priority interrupts are provided to initiate transfers between peripheral devices and the CPU. The higher priority data interrupt, level C_{16}, is used to initiate a data word or byte transfer. The service interrupt, level D_{16}, is used to initiate service routines for end of record, error, and similar signals. Under program control, each I/O priority interrupt can be connected or disconnected within each peripheral device. If a peripheral device has a stored interrupt request when a command is issued to disconnect the interrupt, the request will be reset. No interrupt signals are stored in a disconnected controller. Therefore when a controller is reconnected, all interrupt signals are cleared.

Even though all peripheral devices share the two standard I/O priority interrupts, the party line system used provides rapid response to interrupt requests. When a peripheral device requires a data transfer, the data request flip-flop in the device controller is set. Since the data request line is common to all peripheral devices, any data request flip-flop that is set will cause the data request line to be true. If no higher priority interrupt level is active, the CPU I/O subsystem will issue a data queue update command. In response to this command, all peripheral devices that have their data request flip-flop set, place their priority level on the data lines and their source ID on the source ID lines. The data lines are used to provide priority sub-levels for the data interrupt. The highest sub-level corresponds to data bit 0 on the data lines and the lowest sub-level to data bit 15. During the data queue update command each peripheral device examines all of the data lines corresponding to a higher priority sub-level than its own. If a higher priority sub-level is detected, the peripheral device removes its source ID from the source ID lines. At the end of the data queue update command the following occur:

- The CPU internally stores the source ID of the highest priority peripheral device, to be used for defining the interrupt entry location.
- The highest priority peripheral device resets its data request flip-flop and removes its source ID from the source ID Bus.
- All peripheral devices remove their priority sub-levels from the data lines.

Next, the CPU stores the current contents of the Program Register in location 38_{16} and branches to the address contained in one of sixty-four dedicated locations $(80\text{-}BF)_{16}$ specified by the source ID. The subroutine is then entered to transfer data.

The service interrupt operates in the same manner as the data interrupt, except the dedicated return location is $3A_{16}$ and the dedicated entry locations are $C0_{16}$-FF_{16}.

Refer to Section IV for more information on the I/O interrupts.

DIRECT MEMORY PROCESSOR

The DMP provides direct memory access capability for 8 peripheral device controllers. All 8 controllers can perform transfers of blocks of data to/from computer memory at the same time on an inter-leaved basis. Devices connected to DMP channels also accept Input Data and Output Data commands when not performing DMP controlled block transfers.

A pair of dedicated memory locations are assigned for each of the 8 controllers. Each controller is assigned a Transfer Count (TC) location $(60\text{-}67)_{16}$ and a Transfer Address (TA) location $(70\text{-}77)_{16}$ having the formats:

DMP TRANSFER PARAMETER FORMATS

TRANSFER COUNT

C*	S	NEGATIVE WORD COUNT ≤16384
0		15

*C = 1 Transfer Single Block
C = 0 Transfer Chain of Blocks

TRANSFER ADDRESS

ADDRESS FIELD
0 … 15

Transfer Initiation

Once a peripheral device is appropriately selected and initialized, a data transfer is started by storing the desired starting address for the transfer in the TA location and the negative number of words to be transferred in the TC location. An output command instruction in the transfer initiate format is then executed. Transfers occur automatically at the rate requested by the device. The TA and negative TC are incremented after each transfer. The maximum length of a single block is 16384 words.

When TC equals zero, a data interrupt is generated. If this interrupt is connected by the program to the interrupt (level C_{16}) party line and if the level is enabled, the computer will be interrupted as soon as the data level reaches the top of the interrupt queue.

When the device can accept a new command, the service interrupt (level D_{16}) is generated, if program connected to the service interrupt party line.

The use of both the data and service interrupts provides a choice between two "end of block" signals. One occurs as soon as the last word has been transferred and the other occurs when the device is ready to be commanded again, which is often milliseconds after the last transfer.

Data Chaining

If bit 0 of the Transfer Count is set to zero (C=0) before the initiate command is executed, a new block of words will be transferred automatically after the transfer

of the current block is completed. The data interrupt signifies the completion of each block. The TA and TC parameters for the new block are obtained from the two memory locations immediately following those occupied by the current block. TC is taken from the first location and TA from the second location after the data block.

If C=0 in the TC parameter, data chaining will continue until a TC parameter is encountered with C=1.

Register File

All eight DMP channels are supplied with a pair of registers in which the current TA and TC are stored. Each time a block transfer is initiated, the contents of the TA and TC dedicated memory locations are automatically transferred to the two registers associated with the channel. Transfers can be made over these channels at rates up to 416K (input) or 340K (output) words per second, which are determined by the I/O subsystem timing.

At the end of a transfer sequence, just prior to SI generation, the final TA is stored in a dedicated location from the appropriate channel register.

PERIPHERAL DEVICE ASSIGNMENTS

All programming parameters for MODCOMP peripheral devices are listed in Table 5-1. Unassigned dedicated locations have been left for other peripheral devices, analog input subsystems, communication subsystems, custom devices and future system expansion.

PROGRAMMING CONSIDERATIONS

The sequence of programming steps necessary to perform typical input/output functions are described in this section. The descriptions are general purpose and therefore are designed to cover all contingencies.

REGISTER I/O INTERRUPT MODE SEQUENCE

New Command Initiation:

- Store interrupt subroutine starting addresses in the data and service interrupt level dedicated locations.
- Reset previous error and interrupt status by the execution of an Output Command instruction with a No Op output command, disconnecting both interrupts.
- Test present device status by executing an Input Status instruction.
- If status indicates inoperability or an invalid (all zero) status word, exit to error routine; otherwise continue.

I/O PRIORITY	INTERRUPT LOCATIONS DATA	SERVICE	DMP LOCATIONS TC	TA	DEVICE ADDRESS	PERIPHERAL DEVICE
0	81	C1	61	71	01	Moving Head Disc
1	82	C2	62	72	02	Fixed Head Disc
2						High Level Analog Input Subsystem
	90		60	70	10	-Channel Output
	91	D1	63	73	11	-Data Input
3						Communications Multip.
	98	D8	6F	7F	18	-Controller
	99	D9			19	-Channels
4	83	C3	63	73	03	High Performance Magnetic Tape
5						Wide Range Analog Input System
	92		65	75	12	-Channel Output
	93	D3	66	76	13	-Data Input
6	A0-A7	E0-E7			20-27	Input/Output Interface Subsystem
7	84	C4	64	74	04	Moderate Performance Magnetic Tape
8	85	C5			05	Card Readers (300 and 1,000 CPM)
9	86	C6			06	Card Punch
10	94	D4			14	Wide Range Relay Analog Input Subsystem
11	87	C7			07	Line Printer (600 LPM)
12	88	C8			08	X-Y Plotter
13	89	C9			09	Paper Tape Punch
14	8B	CB			0B	Line Printer (50-150LPM)
15	8A	CA			0A	Teletype/Paper Tape Reader
16	A8-AF	E8-EF			28-2F	Input/Output Interface Subsystem

TABLE 5-1 Peripheral Device Interrupt Assignments

- Execute an output Command Instruction specifying register mode, input or output, connection of interrupts and any other modifiers required by the particular peripheral device. Exit and wait for interrupt.

The controller is now busy and will not respond to new initiation commands. It will respond to Input Status, Input Data and Output Data Instructions and an Output Command Instruction with a Terminate Command. The controller will produce the data interrupt when a data transfer is required and the service interrupt if a malfunction occurs or at the end of the media record.

Response to Data Interrupt:

- The data interrupt processing routine is automatically entered when the requesting controller has the highest priority.
- Preserve original contents of R1 - execute an STM, R1,A. Repeat for all other registers to be used as working registers.
- Check word count, if transfer not complete, perform input or output operation as required. If output, load new data into appropriate place in register, execute Output Data Instruction and update word and byte counts appropriately. If input, execute Input Data Instruction and move or store data as required before updating word and byte counts.
- If the last word required was transferred, an Output Command Instruction should be executed, issuing a Terminate Command to the controller. This will stop further transfers and reset the data interrupt request.
- Restore the previous contents of the working registers.
- Execute a CIR Instruction to exit the routine and return to the original program.

Response to Service Interrupt:

- The Service Interrupt Processing routine is automatically entered if the requesting controller has the highest priority. This interrupt is generated after all hardware checks are complete and the controller can accept a new initiation command.
- Preserve the original contents of R1 - execute an STM,R1,A. Repeat for all registers to be used as working registers.
- Check validity of the transfer by issuing an Input Status instruction. If an abnormality is indicated, exit to error recovery routine.
- If previous checks are satisfactory and no further tasks required for controller, restore the previous contents of the working registers and execute a CIR.
- If previous checks are satisfactory and another transfer sequence is desired, execute an Output Command Instruction with a new initiation command. This command will reset any status conditions that may be set. Execute a CIR to exit the subroutine.

REGISTER I/O TEST AND TRANSFER MODE

Register I/O transfers may be accomplished without the use of data interrupts. A "Data Ready" bit is provided in the standard status word for this purpose. To operate in this mode, the data interrupt is disconnected by the initiation command and the data ready bit is tested during each transfer sequence. Device control is performed in the same manner as in the interrupt mode.

DIRECT MEMORY PROCESSOR I/O MODE

The optional DMP mode frees the program from the task of handling individual data word transfers, and increases net throughput capabilities. The software initiation and termination sequences are described in this Section in the most general manner possible.

<u>New Command Initiation</u>:

- Store interrupt subroutine starting addresses in the two dedicated interrupt level locations.
- Reset previous error or interrupt status by execution of an Output Command, which also disconnects both interrupt levels.
- Test present status by executing an Input Status Instruction. If inoperability is indicated or an invalid (all zero) status word, exit to error routine.
- Store a transfer address and a word count in the two DMP dedicated locations.
- Execute an Output Command Instruction with an Initiate Command specifying DMP mode, input or output, connection of service interrupt, optional connection of data interrupt, and any other modifiers required by the particular peripheral.
- Exit and wait for the interrupt.

The controller is now busy and will not respond to new initiation commands. It will respond only to an Input Status Instruction or an Output Command Instruction with a Terminate or No Op Command.

<u>Response to Data Interrupt</u>:

- The data interrupt processing routine is automatically entered when the controller requesting has highest priority.
- Preserve the original contents of R1 - execute an STM,R1,A. Repeat as required for all working registers.
- The occurrence of this interrupt, in this mode, designates that the transfer of the block of data has been completed (TC=0). The program may use this fact to gain time to manipulate data prior to the completion of the physical media operation.
- Execute a CIR, which will return control to the point of interruption.

Response to Service Interrupt:

- The service interrupt processing routine is automatically entered when the requesting controller is the highest in the interrupt queue. This interrupt is generated after all hardware status checks are complete and the controller is ready to accept a new initiation command.
- Preserve the original contents of R1 - execute an STM,R1,A. Repeat for all other registers to be used as working registers.
- Check validity of the transfer by executing an Input Status instruction. If an abnormality is indicated, exit to an error recovery routine.
- Check final transfer address in dedicated location. If improper, exit to error routine.
- If the previous checks are satisfactory and no further tasks are required, execute a CIR instruction to return to the original program.
- If the previous checks are satisfactory and another block transfer is desired, load the word count and transfer address into the dedicated locations. Then execute an Output Command Instruction with a new Initiation Command. This command will reset any status conditions that may be set.
- Execute a CIR instruction to exit the routine.

OUTPUT COMMAND FORMATS

An Output Command Instruction transfers the 16 bit output command stored in register Ra to the I/O register where it is placed on the I/O bus data lines. There are three basic command formats; Select, Control and Transfer Initiate. The bit designations for each group are defined below. All standard peripheral controllers follow these format conventions. All Commands except End-of-Block and Terminate reset all stored status if the device is not busy. The specific commands and tests for each peripheral device are listed in Appendix C. The standard controllers interpret the command as follows:

Select Format

0	1	2 – 15
0	0	

BITS	FUNCTION
0,1	Must both be zero. These bits specify the select format.
2-15	Specify a set up condition such as unit number (multi-unit controllers), density, head number, etc.

Control Format

0	1	2	3	4	5	6	7	8–15
0	1	D	S	E	T		MPE	

BIT FUNCTION

0 Must be zero.

1 Must be one. This bit is used in conjunction with bit 0 to specify the control format.

2 Specifies the state of the data interrupt:

Zero - Disconnects the device controller and resets the request in the controller if present.

One - Connects the device controller sub-level to the data interrupt level.

If the DMP mode had previously been specified by a Transfer Initiate command, the data interrupt will occur when the Word Count = 0.

If the register I/O mode had previously been specified by a Transfer Initiate command, the data interrupt is defined as Data Request.

3 Specifies the state of the service interrupt.

Zero - Disconnects and resets the request if active.

One - Connect the interrupt, allowing it to become active.

The service interrupt may be caused by a variety of conditions such as end of record or error. The interrupt condition depends on controller design.

4 Specifies End-of-Block command when equal to one. No effect when equal to zero. The End-of-Block command causes the controller (except the Teletype Controller) to immediately generate a data interrupt if that interrupt had previously been connected. This function is useful for diagnostic and debugging purposes. An End-of-Block command will be accepted even when a controller is busy or operating in the DMP mode.

The responding device ignores all bits of the control format except 0, 1, 4 and 5.

5 Specifies Terminate command when equal to one. No effect when equal to zero. The Terminate command stops data transfer to/from the specified device and resets any non-active data interrupt, or DMP Data request. A Terminate command will be accepted when a controller is busy or in the DMP mode.

The Terminate command will also condition a controller to generate a service interrupt when the controller is subsequently ready to respond to another Transfer Initiate command. If the controller is not busy and the service interrupt has been previously connected, this interrupt will occur immediately. The responding device will ignore all bits of the control format except 0, 1, 4 and 5.

	If a terminate command is issued with bit 7 set to one, a controller with DMP facilities will set its memory parity error status indicator. This command is normally issued automatically by the DMP I/O system if such an error is detected.
	If bit 7 is set to one within a terminate command to some devices (TTY for example), an immediate operation abort occurs.
6	Normally used to distinguish between a normal control command and a No-Op. See No-Op below.
7*	Specify a control function such as rewind, advance record, seek cylinder, etc..

<u>No Op Command</u>

When bits 4, 5 and 6 are all zero, bits 7-15 are ignored. The No Op command alters interrupt connection per the values of bits 2 and 3 whether or not the device is busy. The No Op command also resets all device status if the device is not busy.

<u>Interrupt Disconnection and Termination</u>

An interrupt may be reset by means of a Disconnect or Terminate command whenever the interrupt level is Active. However, if the level is not active but might become active immediately, an invalid request might occur on the I/O level. To accommodate this situation, requests at levels 80 and C0 should execute a CIR and return to the interrupted program.

<u>Transfer Initiate</u>

0	1	2	3	4	5 — 15
1	M	D	S	I	

<u>BITS</u>	<u>FUNCTION</u>
0	Must be one. This bit specifies the Transfer Initiate Format.
1	Specifies mode selection for subsequent data transfer.
	Zero - Sets the device to the programmed register I/O mode. The device sends a data interrupt request, if connected, each time it requires a data transfer, including the first transfer.
	One - Sets the device to the DMP transfer mode.
2	Specifies the state of the data interrupt:
	Zero - Disconnects the device controller and resets the request in the controller if present.
	One - Connects the device controller sub-level to the data interrupt level.
	If the DMP mode had previously been specified by a Transfer Initiate command, the data interrupt will occur when the TC = 0.
	If the register I/O mode had previously been specified by a Transfer Initiate command, the data interrupt is defined as Data Request.

*Except as already noted.

3 Specifies the state of the service interrupt.

Zero - Disconnects and resets the request if active.

One - Connects the interrupt, allowing it to become active.

The service interrupt may be caused by a variety of conditions such as the end of the record or error. The interrupt condition depends on controller design.

4 Specifies the direction of data transfer.

Zero - Sets the device to the output transfer mode.

One - Sets the device to the input transfer mode.

5-15 Specifies a transfer initiate function such as write record or read card. (See Appendix C).

INPUT STATUS FORMAT

An Input Status instruction causes the contents of the 16 data lines to be transferred to the specified register Ra. The controller, as selected by the device address, puts its status word on the data lines and then the transfer occurs.

One basic format exists which is common to all controllers. This format encompasses two groups: Errors and Events. The error group has a pointer bit indicating if any error is set. The status format is so defined that a status word of all zeros is invalid, indicating a malfunctioning or non-existant controller.

0	1	2	3	4	5	6	7	8	9	15
E	D	P	I	M			B	T		

$\underbrace{\text{Bits 0–6}}_{\text{ERROR FIELD}}$ $\underbrace{\text{Bits 7–15}}_{\text{EVENT FIELD}}$

ERROR FIELD

EVENT FIELD

BIT	STATUS
0	Error Pointer Bit Zero - An error has occurred and is defined in the field of bits 1 through 6. One - No error has occurred.

BIT	STATUS
1	Data transfer error. Zero - No error. One - Overflow or underflow error.
2	Parity or checksum error. Zero - No error. One - Device parity error.

3 Inoperable.

Zero - Device operable (on-line).

One - Device inoperable (off-line, interlock open, etc.)

4 Memory parity error.

Zero - No error.

One - A memory parity error was detected during a DMP transfer.

5-6 Specify error conditions unique to a device such as seek error.

7 Busy status of device controller.

Zero = Device controller not busy.

One - Device controller busy.

8 Transfer Status. (Normally used when not operating in the interrupt mode).

Zero - Device controller ready to transfer a data word.

One - Device controller not ready to transfer a data word.

9-15 Specify device unique event conditions.

I/O BUS INTERFACE

The MODCOMP II I/O Bus is a time-shared (party line) bi-directional bus capable of transferring data, commands, and device status. Since the bus is a party line, each device controller must request use of the bus on one or more of the three request lines available. When the bus is available for use, the CPU will issue interlock signals which inform the specified controller when it may use the bus.

The I/O cable is connected serially (daisy chained) to each Peripheral Controller Interface and Input Output Interface Subsystem. All controllers and the CPU are connected to the I/O bus through a cable driver/receiver set which isolates the logic from the effect of I/O bus loading and delays.

The I/O cable connection rules are:

(1) The maximum combined cable length per computer is 100 feet.
(2) All connections must be made via a cable driver/receiver set.
(3) A maximum of eight cable driver/receiver sets can be connected to the cable.
(4) Up to four device controllers can be connected to each driver/receiver set.

The block diagram of the computer I/O subsystem (Figure 5-2) shows the types of signals in the I/O cable (which is cut by the dashed line in the figure). Thirty-nine pairs of lines are provided for communication between the CPU and external controllers.

The signals are described below.

Address (6 pairs DA00N through DA05N) - Designates which device controller is to respond to the control lines. The six-bit binary value is obtained from the I/O instruction word (bits 6-7, 12-15).

Control (5 pairs) - The four I/O instruction functions are coded on two lines. (DACSN) - Command (false) / Data (true) and (DAION) - Input (true) / Output (false).

The I/O Sync signal (IOISN) indicates that an I/O transfer is occurring. When this signal is true, the Command/Data, Input/Output and Address signals are valid.

The other two control signals are Master Clear (IOICB), which is generated at the computer, and Clock (IOCLKN), which is a 5 MHZ square wave generated in the computer. The Master Clear signal normalizes all units and the Clock signal provides a timing signal to controllers, eliminating the need for an internal time base.

Data (16 pairs IOB00N through IOB15N) - All commands, status, data and controller priority are transmitted over this bi-directional bus.

DMP Control (2 pairs) - The DMP Request signal (DMRQN) is sent to the computer by controllers operating under DMP Control. It signifies a request for a word transfer in the direction the DMP channel was last initialized.

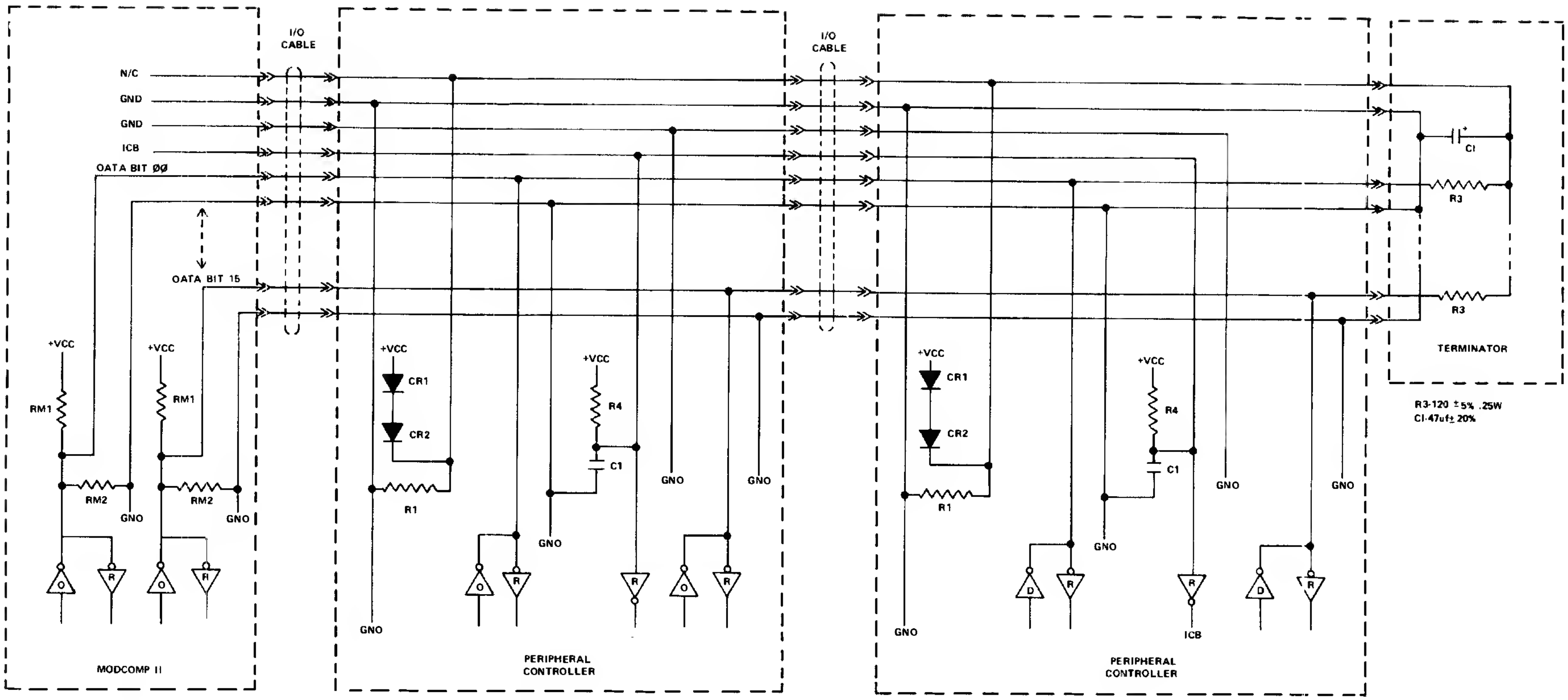
The DMP responds with a DMP Queue Update signal (UDDMQN). In response to this signal, each (if more than one) controller having a current DMP request places a true signal on one of the 16 data bus lines, indicating one of 16 DMP channel priorities.*

The Source ID of the unit, which is a six-bit pointer to the dedicated registers for the channel is also placed on the Source ID lines. Each requesting unit examines the data bus lines to determine if a higher priority unit is also requesting a transfer. If so, the lower priority unit removes its priority and Source ID signals.

Priority Interrupt Control (4 pairs) - These signals consist of a Request signal and an Update Queue signal for both of the party line interrupts. Therefore the four signals are Data Request (DIRQN), Data Queue Update (UDDIQN), Service Request (SIRQN), and Service Queue Update (UDSIQN). These two pairs of signals are handled just like the DMP Request and DMP Queue Update except they are serviced at priority interrupt levels C_{16} and D_{16}, and DMP Transfers are serviced in the DMP independently of the interrupt priority structure. Each requesting controller places its priority bit on one of the 16 data bus lines and its Source ID on the Source ID lines. All but the highest priority requesting controller remove their signals during the Update Queue signal before the Source ID signals are transferred to the computer.

Source ID (6 pairs SID00N through SID05N) - Each device has a unique Source ID code which is sent to the computer in response to an update queue command. This code is used to identify the dedicated memory location assigned to the interrupt sublevel or the DMP channel.

* Although 16 channel priorities are available, only eight (8) may be used.



R3-120 ±5% .25W
CI-47uf± 20%

RM1-220 Ω ±3%, 140MW

RM2-330 Ω ±3%, 140MW

D-SINKS 70 MA MIN. FOR
LOGICAL 1 (.4V MAX.)

-100μA MAX. LEAKAGE
FOR A LOGICAL Ø (5V MAX.)

R-Ø OUT FOR INPUT≥2.0V
-1 OUT FOR INPUT≤.8V
MAX SOURCE CURRENT-2MA

R1-220 Ω ±5%, .25W

C1-.01μF, CERAMIC
(-20%, +80%)

CR1, CR2-IN4001 or EQUIV.

R4-1.2K±5%, .25W

R1-220 Ω ±5%, .25W

C1-.01μF, CERAMIC
(-20%+80%)

CR1, CR2-IN4001 OR EQUIV.

R4-1.2K±5%, .25W

FIGURE 5-2 INPUT/OUTPUT CABLE OIAGRAM

Signal Levels

The signal levels at the cable driver/receiver interface to the controller logic are:

Logic 0:	2.4 to 5.5 Volts	Except IOICB
Logic 1:	0 to 0.4 Volts	which is inverted.

Figure 5-3 illustrates a MODCOMP II I/O Bus system with the driver/receiver sets and the associated termination network.

The I/O Clock (IOCLKN) a 5 MHz square wave, is distributed on the I/O bus for general purpose use in the controllers.

When the IOICB (Initial Condition Bus) comes true, each controller should normalize (not busy, no interrupt or DMP Requests, no interrupt enabled, device normalized, etc). This signal is present if the CPU power is going down and is present to normalize the system when power is returned to the CPU and when the master clear switch on the CPU console is depressed.

All signals are ground true except IOICB. This allows each controller to recognize the absence of a current sink in the CPU and normalize when CPU power is absent.

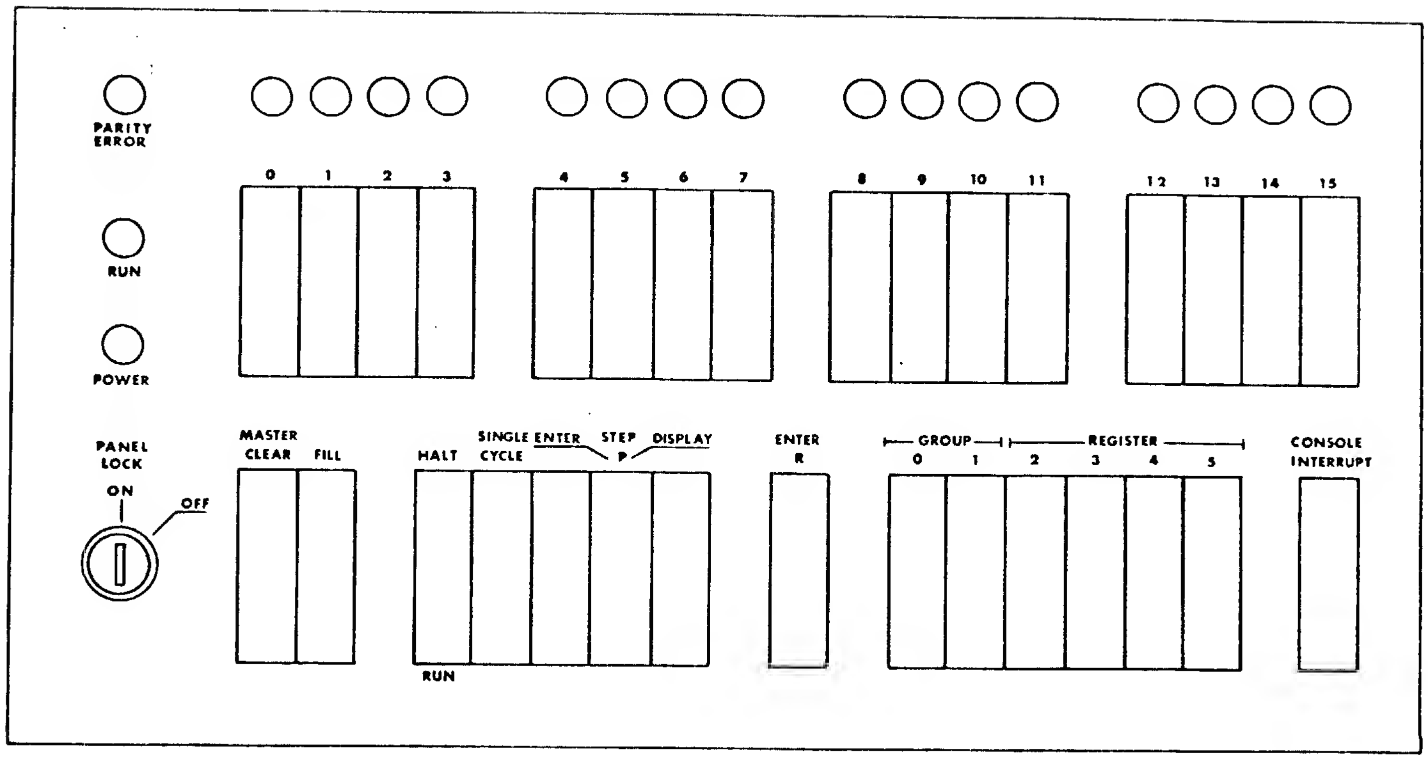


Figure 6-1 Control Panel

VI. OPERATOR CONTROLS

The MODCOMP II control panel enables programs to be loaded into memory and executed under manual control. It also provides a number of debugging and maintenance aids.

INDICATORS

Data

The 16 Data Indicators display the contents of the register designated by the Register Select switches when the computer is halted. The bus traffic is displayed when the computer is in the run mode.

Parity Error

This indicator is lighted when the computer is halted due to a parity error, if no System Protect Feature, or until the memory parity interrupt is serviced, if the System Protect Feature is present.

Run

This indicator is lighted when the computer is in the run mode, which means not halted manually or by execution of the Halt instruction.

Power On

This indicator is lighted when a-c power is applied to the computer. The circuit breaker for switching power is located behind a hinged panel in the top front of the system cabinet which contains the computer.

SWITCHES

Data Entry

The 16 Data Entry switches are used to enter data into any register or memory location. The lowered position corresponds to a one value and the raised (normal) position corresponds to a zero value.

Panel Lock

In the ON position; this keyswitch disables all other control panel switches except the 16 Data Switches and the Console Interrupt switch. In the ON position it also enables the System Protect operation. All switches are enabled and the System Protect is disabled when the Panel Lock switch is in the OFF position.

Master Clear

Depressing this switch causes the computer and peripheral devices to be cleared. All interrupts and control signals and the contents of the Program and Instruction Registers are reset to the zero or cleared state.

Fill

Depressing this switch causes a bootstrap routine to be transferred to main memory (locations 0-$2D_{16}$) from read-only memory. The bootstrap routine automatically fills from the selected device. The device is selected by setting the proper device number in the Data Entry switches prior to depressing the Fill switch:

Fill From	Data Entry Switches_{16}
Paper Tape Reader	0 0 0 A
Card Reader	0 0 0 5
Mag tape	0 0 0 4
Fixed Head Disc	0 0 0 2
Moving Head Disc	0 0 0 1

Run/Halt

This switch is used to manually place the computer in either of the modes indicated by the switch positions. When the computer is manually halted, the Program Register points to the next instruction and the Instruction Register contains this next instruction. To resume operation at a new location, the Master Clear switch should be depressed to clear the Instruction Register, and the new location minus one should be manually entered into the Program Register. The Halt/Run switch should then be raised to the Run position.

Single Cycle

Depressing this switch causes the instruction presently stored in the Instruction Register to be executed. The Program Register is then advanced to the next instruction, and this instruction is accessed from memory and transferred to the Memory Data and Instruction Register. It can be displayed from the Memory Data Register.

Enter

When this switch is depressed, the word corresponding to the position of the Data Entry switches is stored in the memory location specified by the contents of the Program Register. The Program Register is not advanced.

Step P

The contents of the Program Register are incremented by one and the contents of the new memory location are entered into the MDR when this switch is depressed. The switch is provided to facilitate modifying or displaying the contents of consecutive memory locations.

Console Interrupt

Depressing this switch, in computer models having the Executive Features, causes an interrupt request signal to be sent to interrupt Level E.

Display

The contents of the memory location designated by the contents of the Program Register are displayed and entered into IR when this switch is depressed, providing the Register Select switches designate the Memory Data Register.

Enter R

Depressing this switch causes the contents of the Data Entry switches to be stored in the register specified by the Register Select switches.

Register Select

These switches are used to specify the register, the contents of which are to be displayed or modified. Whenever the computer is halted, the Data indicators display the contents of the specified register. When the Enter R switch is depressed, the specified register contents are replaced by the word specified by the Data Entry switches.

SOURCE (Display Register)		DESTINATION (Enter Register)	
Switch Setting	Register Name	Switch Setting	Register Name
ØØ - ØF	- GPR FILE	ØØ	- NO DEST.
1Ø - 17	- UNASSIGNED	Ø1 - ØF	- GPR FILE
18	- NO SOURCE	1Ø	- AR
19 - 1F	- GPR 1-7	11	- AR & TRB
2Ø - 2F	- AUX FILE	12, 13	- UNASSIGNED
3Ø	- ACTIVE	14	- Ra → PAUSE COUNTER
31	- TRB	15	- Rb → PAUSE COUNTER
32	- OPT PLØ	16	- BUS → PAUSE COUNTER
33	- OPT PL1	17	- $\overline{\text{Rb}}$ → PAUSE COUNTER
34	- ENABLE	18	- HLT FF
35	- TRA	19	- RMI FF
36	- P.I. QUEUE	1A, 1B	- UNASSIGNED
37	- LITERAL	1C	- ROM ADDRESS REG.
38	- REQUEST	1D	- UBR
39	- UNASSIGNED	1E	- LBR
3A	- INPUT BUFFER	1F	- LBR, ZUBR
3B	- UNASSIGNED	2Ø - 2F	- AUX FILE
3C	- PR	3Ø	- ACTIVE
3D	- OFLO (BITØ)	31	- TRB
	CARRY SAVE (BIT 15)	32	- OPT PLØ
3E	- UNASSIGNED	33	- OPT PL1
3F	- MDR	34	- ENABLE
		35	- TRA
		36	- MA
		37	- MA & TRA
		38	- REQUEST
		39	- TRA & TRB
		3A	- OUTPUT BUFFER
		3B	- UNASSIGNED
		3C	- P & MA & TRA
		3D	- LSH MDR
		3E	- MSH MDR
		3F	- BOTH MDR

TABLE 6-1 REGISTER SELECT

CONTROL PANEL OPERATION

DISPLAY REGISTER

1. HLT/RUN switch to HLT
2. Register select switches to the desired register (Lights display contents of register)

LOAD REGISTER

1. HLT/RUN switch to HLT
2. Register select switches to the desired register
3. Place data into R0 (switch register)
4. Press 'ENTER REGISTER'

LOAD MEMORY

1. HLT/RUN switch to HLT
2. Load R11 (PR) with desired starting address
3. Load R0 (switch register) with desired data
4. Press 'ENTER MEMORY'

Note: To load sequential locations, press 'STEP' one time, and repeat steps 3 and 4.

DISPLAY MEMORY

1. HLT/RUN switch to HLT
2. Load R11 (PR) with desired starting address
3. Set the register select switches to R17
4. Press 'DISPLAY MEMORY' (Lights will display the contents of the selected memory location)

Note: To display sequential locations, press 'STEP' switch for each additional location to be displayed.

START PROGRAM

1. HLT/RUN switch to HLT
2. Load R11 (PR) with desired starting address
3. Press 'DISPLAY MEMORY'
4. HLT/RUN switch to RUN

SINGLE CYCLE PROGRAM

1. HLT/RUN switch to HLT
2. Register select switches to R17 (MDR)
3. Press 'DISPLAY MEMORY'
4. Press 'SINGLE CYCLE' for each instruction to be executed. R17 will display the contents of the first word of the next instruction to be executed.

Note: Interrupts will be ignored during single cycle.

FILL

1. HLT/RUN switch to HLT
2. R0 bits 12-15 to the device address of filling device
   A. 1 - moving head disc
   B. 2 - fixed head disc
   C. 4 - mag tape
   D. 5 - card reader
   E. A - TTY or paper tape
3. Mag tape only

   R0 bits 1-7 to file for mag tape fill

   Disc only

   R0 bits 1-7 to $\frac{\text{Starting Sector}}{\#100}$ (Starting Sector divided by 100)
4. Press 'MASTER CLEAR'
5. Press 'FILL'
6. HLT/RUN switch to RUN

APPENDIX A. HEXADECIMAL TO DECIMAL CONVERSION

This appendix enables direct conversion of decimal numbers to/from hexadecimal numbers in the ranges:

HEXADECIMAL	DECIMAL
000 to FFF	0000 to 4095

For numbers outside the range of the table, add the following values to the table figures:

HEXADECIMAL	DECIMAL	HEXADECIMAL	DECIMAL
1000	4096	9000	36864
2000	8192	A000	40960
3000	12288	B000	45056
4000	16384	C000	49152
5000	20480	D000	53248
6000	24576	E000	57344
7000	28672	F000	61440
8000	32768		

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013	0014	0015
010	0016	0017	0018	0019	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031
020	0032	0033	0034	0035	0036	0037	0038	0039	0040	0041	0042	0043	0044	0045	0046	0047
030	0048	0049	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	0060	0061	0062	0063
040	0064	0065	0066	0067	0068	0069	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079
050	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	0090	0091	0092	0093	0094	0095
060	0096	0097	0098	0099	0100	0101	0102	0103	0104	0105	0106	0107	0108	0109	0110	0111
070	0112	0113	0114	0115	0116	0117	0118	0119	0120	0121	0122	0123	0124	0125	0126	0127
080	0128	0129	0130	0131	0132	0133	0134	0135	0136	0137	0138	0139	0140	0141	0142	0143
090	0144	0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159
0A0	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	0170	0171	0172	0173	0174	0175
0B0	0176	0177	0178	0179	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191
0C0	0192	0193	0194	0195	0196	0197	0198	0199	0200	0201	0202	0203	0204	0205	0206	0207
0D0	0208	0209	0210	0211	0212	0213	0214	0215	0216	0217	0218	0219	0220	0221	0222	0223
0E0	0224	0225	0226	0227	0228	0229	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239
0F0	0240	0241	0242	0243	0244	0245	0246	0247	0248	0249	0250	0251	0252	0253	0254	0255
100	0256	0257	0258	0259	0260	0261	0262	0263	0264	0265	0266	0267	0268	0269	0270	0271
110	0272	0273	0274	0275	0276	0277	0278	0279	0280	0281	0282	0283	0284	0285	0286	0287
120	0288	0289	0290	0291	0292	0293	0294	0295	0296	0297	0298	0299	0300	0301	0302	0303
130	0304	0305	0306	0307	0308	0309	0310	0311	0312	0313	0314	0315	0316	0317	0318	0319
140	0320	0321	0322	0323	0324	0325	0326	0327	0328	0329	0330	0331	0332	0333	0334	0335
150	0336	0337	0338	0339	0340	0341	0342	0343	0344	0345	0346	0347	0348	0349	0350	0351
160	0352	0353	0354	0355	0356	0357	0358	0359	0360	0361	0362	0363	0364	0365	0366	0367
170	0368	0369	0370	0371	0372	0373	0374	0375	0376	0377	0378	0379	0380	0381	0382	0383
180	0384	0385	0386	0387	0388	0389	0390	0391	0392	0393	0394	0395	0396	0397	0398	0399
190	0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	0410	0411	0412	0413	0414	0415
1A0	0416	0417	0418	0419	0420	0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431
1B0	0432	0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444	0445	0446	0447
1C0	0448	0449	0450	0451	0452	0453	0454	0455	0456	0457	0458	0459	0460	0461	0462	0463
1D0	0464	0465	0466	0467	0468	0469	0470	0471	0472	0473	0474	0475	0476	0477	0478	0479
1E0	0480	0481	0482	0483	0484	0485	0486	0487	0488	0489	0490	0491	0492	0493	0494	0495
1F0	0496	0497	0498	0499	0500	0501	0502	0503	0504	0505	0506	0507	0508	0509	0510	0511

200	0512	0513	0514	0515	0516	0517	0518	0519	0520	0521	0522	0523	0524	0525	0526	0527
210	0528	0529	0530	0531	0532	0533	0534	0535	0536	0537	0538	0539	0540	0541	0542	0543
220	0544	0545	0546	0547	0548	0549	0550	0551	0552	0553	0554	0555	0556	0557	0558	0559
230	0560	0561	0562	0563	0564	0565	0566	0567	0568	0569	0570	0571	0572	0573	0574	0575
240	0576	0577	0578	0579	0580	0581	0582	0583	0584	0585	0586	0587	0588	0589	0590	0591
250	0592	0593	0594	0595	0596	0597	0598	0599	0600	0601	0602	0603	0604	0605	0606	0607
260	0608	0609	0610	0611	0612	0613	0614	0615	0616	0617	0618	0619	0620	0621	0622	0623
270	0624	0625	0626	0627	0628	0629	0630	0631	0632	0633	0634	0635	0636	0637	0638	0639
280	0640	0641	0642	0643	0644	0645	0646	0647	0648	0649	0650	0651	0652	0653	0654	0655
290	0656	0657	0658	0659	0660	0661	0662	0663	0664	0665	0666	0667	0668	0669	0670	0671
2A0	0672	0673	0674	0675	0676	0677	0678	0679	0680	0681	0682	0683	0684	0685	0686	0687
2B0	0688	0689	0690	0691	0692	0693	0694	0695	0696	0697	0698	0699	0700	0701	0702	0703
2C0	0704	0705	0706	0707	0708	0709	0710	0711	0712	0713	0714	0715	0716	0717	0718	0719
2D0	0720	0721	0722	0723	0724	0725	0726	0727	0728	0729	0730	0731	0732	0733	0734	0735
2E0	0736	0737	0738	0739	0740	0741	0742	0743	0744	0745	0746	0747	0748	0749	0750	0751
2F0	0752	0753	0754	0755	0756	0757	0758	0759	0760	0761	0762	0763	0764	0765	0766	0767
300	0768	0769	0770	0771	0772	0773	0774	0775	0776	0777	0778	0779	0780	0781	0782	0783
310	0784	0785	0786	0787	0788	0789	0790	0791	0792	0793	0794	0795	0796	0797	0798	0799
320	0800	0801	0802	0803	0804	0805	0806	0807	0808	0809	0810	0811	0812	0813	0814	0815
330	0816	0817	0818	0819	0820	0821	0822	0823	0824	0825	0826	0827	0828	0829	0830	0831
340	0832	0833	0834	0835	0836	0837	0838	0839	0840	0841	0842	0843	0844	0845	0846	0847
350	0848	0849	0850	0851	0852	0853	0854	0855	0856	0857	0858	0859	0860	0861	0862	0863
360	0864	0865	0866	0867	0868	0869	0870	0871	0872	0873	0874	0875	0876	0877	0878	0879
370	0880	0881	0882	0883	0884	0885	0886	0887	0888	0889	0890	0891	0892	0893	0894	0895
380	0896	0897	0898	0899	0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	0910	0911
390	0912	0913	0914	0915	0916	0917	0918	0919	0920	0921	0922	0923	0924	0925	0926	0927
3A0	0928	0929	0930	0931	0932	0933	0934	0935	0936	0937	0938	0939	0940	0941	0942	0943
3B0	0944	0945	0946	0947	0948	0949	0950	0951	0952	0953	0954	0955	0956	0957	0958	0959
3C0	0960	0961	0962	0963	0964	0965	0966	0967	0968	0969	0970	0971	0972	0973	0974	0975
3D0	0976	0977	0978	0979	0980	0981	0982	0983	0984	0985	0986	0987	0988	0989	0990	0991
3E0	0992	0993	0994	0995	0996	0997	0998	0999	1000	1001	1002	1003	1004	1005	1006	1007
3F0	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
400	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039
410	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055
420	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071
430	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087
440	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103
450	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119
460	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135
470	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151
480	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167
490	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183
4A0	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199
4B0	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215
4C0	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231
4D0	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247
4E0	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263
4F0	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279
500	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295
510	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311
520	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327
530	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343
540	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359
550	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375
560	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391
570	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407
580	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423
590	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439
5A0	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455
5B0	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471
5C0	1472	1473	1474	1475	1476	1477	1478	1779	1480	1481	1482	1483	1484	1485	1486	1487
5D0	1488	1489	1490	1491	1492	1493	1494	1495	1496	1497	1498	1499	1500	1501	1502	1503
5E0	1504	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518	1519
5F0	1520	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
600	1536	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550	1551
610	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565	1566	1567
620	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582	1583
630	1584	1585	1586	1587	1588	1589	1590	1591	1592	1593	1594	1595	1596	1597	1598	1599
640	1600	1601	1602	1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615
650	1616	1617	1618	1619	1620	1621	1622	1623	1624	1625	1626	1627	1628	1629	1630	1631
660	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644	1645	1646	1647
670	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1661	1662	1663
680	1664	1665	1666	1667	1668	1669	1670	1671	1672	1673	1674	1675	1676	1677	1678	1679
690	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695
6A0	1696	1697	1698	1699	1700	1701	1702	1703	1704	1705	1706	1707	1708	1709	1710	1711
6B0	1712	1713	1714	1715	1716	1717	1718	1719	1720	1721	1722	1723	1724	1725	1726	1727
6C0	1728	1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742	1743
6D0	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756	1757	1758	1759
6E0	1760	1761	1762	1763	1764	1765	1766	1767	1768	1769	1770	1771	1772	1773	1774	1775
6F0	1776	1777	1778	1779	1780	1781	1782	1783	1784	1785	1786	1787	1788	1789	1790	1791
700	1792	1793	1794	1795	1796	1797	1798	1799	1800	1801	1802	1803	1804	1805	1806	1807
710	1808	1809	1810	1811	1812	1813	1814	1815	1816	1817	1818	1819	1820	1821	1822	1823
720	1824	1825	1826	1827	1828	1829	1830	1831	1832	1833	1834	1835	1836	1837	1838	1839
730	1840	1841	1842	1843	1844	1845	1846	1847	1848	1849	1850	1851	1852	1853	1854	1855
740	1856	1857	1858	1859	1860	1861	1862	1863	1864	1865	1866	1867	1868	1869	1870	1871
750	1872	1873	1874	1875	1876	1877	1878	1879	1880	1881	1882	1883	1884	1885	1886	1887
760	1888	1889	1890	1891	1892	1893	1894	1895	1896	1897	1898	1899	1900	1901	1902	1903
770	1904	1905	1906	1907	1908	1909	1910	1911	1912	1913	1914	1915	1916	1917	1918	1919
780	1920	1921	1922	1923	1924	1925	1926	1927	1928	1929	1930	1931	1932	1933	1934	1935
790	1936	1937	1938	1939	1940	1941	1942	1943	1944	1945	1946	1947	1948	1949	1950	1951
7A0	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967
7B0	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983
7C0	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
7D0	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
7E0	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031
7F0	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
800	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063
810	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079
820	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095
830	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111
840	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
850	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143
860	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159
870	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175
880	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191
890	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207
8A0	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223
8B0	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239
8C0	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255
8D0	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271
8E0	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287
8F0	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303
900	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319
910	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335
920	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351
930	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367
940	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383
950	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399
960	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415
970	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431
980	2432	2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447
990	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460	2461	2462	2463
9A0	2464	2465	2466	2467	2468	2469	2470	2471	2472	2473	2474	2475	2476	2477	2478	2479
9B0	2480	2481	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493	2494	2495
9C0	2496	2497	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2509	2510	2511
9D0	2512	2513	2514	2515	2516	2517	2518	2519	2520	2521	2522	2523	2524	2525	2526	2527
9E0	2528	2529	2530	2531	2532	2533	2534	2535	2536	2537	2538	2539	2540	2541	2542	2543
9F0	2544	2545	2546	2547	2548	2549	2550	2551	2552	2553	2554	2555	2556	2557	2558	2559

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A00	2560	2561	2562	2563	2564	2565	2566	2567	2568	2569	2570	2571	2572	2573	2574	2575
A10	2576	2577	2578	2579	2580	2581	2582	2583	2584	2585	2586	2587	2588	2589	2590	2591
A20	2592	2593	2594	2595	2596	2597	2598	2599	2600	2601	2602	2603	2604	2605	2606	2607
A30	2608	2609	2610	2611	2612	2613	2614	2615	2616	2617	2618	2619	2620	2621	2622	2623
A40	2624	2625	2626	2627	2628	2629	2630	2631	2632	2633	2634	2635	2636	2637	2638	2639
A50	2640	2641	2642	2643	2644	2645	2646	2647	2648	2649	2650	2651	2652	2653	2654	2655
A60	2656	2657	2658	2569	2660	2661	2662	2663	2664	2665	2666	2667	2668	2669	2670	2671
A70	2672	2673	2674	2675	2676	2677	2678	2679	2680	2681	2682	2683	2684	2685	2686	2687
A80	2688	2689	2690	2691	2692	2693	2694	2695	2696	2697	2698	2699	2700	2701	2702	2703
A90	2704	2705	2706	2707	2708	2709	2710	2711	2712	2713	2714	2715	2716	2717	2718	2719
AA0	2720	2721	2722	2723	2724	2725	2726	2727	2728	2729	2730	2731	2732	2733	2734	2735
AB0	2736	2737	2738	2739	2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	2750	2751
AC0	2752	2753	2754	2755	2756	2757	2758	2759	2760	2761	2762	2763	2764	2765	2766	2767
AD0	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783
AE0	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799
AF0	2800	2801	2802	2803	2804	2805	2806	2807	2808	2809	2810	2811	2812	2813	2814	2815
B00	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831
B10	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847
B20	2848	2849	2850	2851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863
B30	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879
B40	2880	2881	2882	2883	2884	2885	2886	2887	2888	2889	2890	2891	2892	2893	2894	2895
B50	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911
B60	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927
B70	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943
B80	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959
B90	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969	2970	2971	2972	2973	2974	2975
BA0	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991
BB0	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007
BC0	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023
BD0	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039
BE0	3040	3041	3042	3043	3044	3045	3046	3047	3048	3049	3050	3051	3052	3053	3054	3055
BF0	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065	3066	3067	3068	3069	3070	3071

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C00	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087
C10	3088	3089	3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103
C20	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113	3114	3115	3116	3117	3118	3119
C30	3120	3121	3122	3123	3124	3125	3126	3127	3128	3129	3130	3131	3132	3133	3134	3135
C40	3136	3137	3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151
C50	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161	3162	3163	3164	3165	3166	3167
C60	3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183
C70	3184	3185	3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199
C80	3200	3201	3202	3203	3204	3205	3206	3207	3208	3209	3210	3211	3212	3213	3214	3215
C90	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231
CA0	3232	3233	3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247
CB0	3248	3249	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	3260	3261	3262	3263
CC0	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279
CD0	3280	3281	3282	3283	3284	3285	3286	3287	3288	3289	3290	3291	3292	3293	3294	3295
CE0	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310	3311
CF0	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327
D00	3328	3329	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343
D10	3344	3345	3346	3347	3348	3349	3350	3351	3352	3353	3354	3355	3356	3357	3358	3359
D20	3360	3361	3362	3363	3364	3365	3366	3367	3368	3369	3370	3371	3372	3373	3374	3375
D30	3376	3377	3378	3379	3380	3381	3382	3383	3384	3385	3386	3387	3388	3389	3390	3391
D40	3392	3393	3394	3395	3396	3397	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407
D50	3408	3409	3410	3411	3412	3413	3414	3415	3416	3417	3418	3419	3420	3421	3422	3423
D60	3424	3425	3426	3427	3428	3429	3430	3431	3432	3433	3434	3435	3436	3437	3438	3439
D70	3440	3441	3442	3443	3444	3445	3446	3447	3448	3449	3450	3451	3452	3453	3454	3455
D80	3456	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3471
D90	3472	3473	3474	3475	3476	3477	3478	3479	3480	3481	3482	3483	3484	3485	3486	3487
DA0	3488	3489	3490	3491	3492	3493	3494	3495	3496	3497	3498	3499	3500	3501	3502	3503
DB0	3504	3505	3506	3507	3508	3509	3510	3511	3512	3513	3514	3515	3516	3517	3518	3519
DC0	3520	3521	3522	3523	3524	3525	3526	3527	3528	3529	3530	3531	3532	3533	3534	3535
DD0	3536	3537	3538	3539	3540	3541	3542	3543	3544	3545	3546	3547	3548	3549	3550	3551
DE0	3552	3553	3554	3555	3556	3557	3558	3559	3560	3561	3562	3563	3564	3565	3566	3567
DF0	3568	3569	3570	3571	3572	3573	3574	3575	3576	3577	3578	3579	3580	3581	3582	3583

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E00	3584	3585	3586	3587	3588	3589	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599
E10	3600	3601	3602	3603	3604	3605	3606	3607	3608	3609	3610	3611	3612	3613	3614	3615
E20	3616	3617	3618	3619	3620	3621	3622	3623	3624	3625	3626	3627	3628	3629	3630	3631
E30	3632	3633	3634	3635	3636	3637	3638	3639	3640	3641	3642	3643	3644	3645	3646	3647
E40	3648	3649	3650	3651	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663
E50	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678	3679
E60	3680	3681	3682	3683	3684	3685	3686	3687	3688	3689	3690	3691	3692	3693	3694	3695
E70	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708	3709	3710	3711
E80	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
E90	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743
EA0	3744	3745	3746	3747	3748	3749	3750	3751	3752	3753	3754	3755	3756	3757	3758	3759
EB0	3760	3761	3762	3763	3764	3765	3766	3767	3768	3769	3770	3771	3772	3773	3774	3775
EC0	3776	3777	3778	3779	3780	3781	3782	3783	3784	3785	3786	3787	3788	3789	3790	3791
ED0	3792	3793	3794	3795	3796	3797	3798	3799	3800	3801	3802	3803	3804	3805	3806	3807
EE0	3808	3809	3810	3811	3812	3813	3814	3815	3816	3817	3818	3819	3820	3821	3822	3823
EF0	3824	3825	3826	3827	3828	3829	3830	3831	3832	3833	3834	3835	3836	3837	3838	3839
F00	3840	3841	3842	3843	3844	3845	3846	3847	3848	3849	3850	3851	3852	3853	3854	3855
F10	3856	3857	3858	3859	3860	3861	3862	3863	3864	3865	3866	3867	3868	3869	3870	3871
F20	3872	3873	3874	3875	3876	3877	3878	3879	3880	3881	3882	3883	3884	3885	3886	3887
F30	3888	3889	3890	3891	3892	3893	3894	3895	3896	3897	3898	3899	3900	3901	3902	3903
F40	3904	3905	3906	3907	3908	3909	3910	3911	3912	3913	3914	3915	3916	3917	3918	3919
F50	3920	3921	3922	3923	3924	3925	3926	3927	3928	3929	3930	3931	3932	3933	3934	3935
F60	3936	3937	3938	3939	3940	3941	3942	3943	3944	3945	3946	3947	3948	3949	3950	3951
F70	3952	3953	3954	3955	3956	3957	3958	3959	3960	3961	3962	3963	3964	3965	3966	3967
F80	3968	3969	3970	3971	3972	3973	3974	3975	3976	3977	3978	3979	3980	3981	3982	3983
F90	3984	3985	3986	3987	3988	3989	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
FA0	4000	4001	4002	4003	4004	4005	4006	4007	4008	4009	4010	4011	4012	4013	4014	4015
FB0	4016	4017	4018	4019	4020	4021	4022	4023	4024	4025	4026	4027	4028	4029	4030	4031
FC0	4032	4033	4034	4035	4036	4037	4038	4039	4040	4041	4042	4043	4044	4045	4046	4047
FD0	4048	4049	4050	4051	4052	4053	4054	4055	4056	4057	4058	4059	4060	4061	4062	4063
FE0	4064	4065	4066	4067	4068	4069	4070	4071	4072	4073	4074	4075	4076	4077	4078	4079
FF0	4080	4081	4082	4083	4084	4085	4086	4087	4088	4089	4090	4091	4092	4093	4094	4095

APPENDIX B. CHARACTER CODES

CHAR.	CODE	L.P.	CODE	CARD READER	CODE 029	TTY	CODE	EBCDIC	CODE	COMP. CHAR.	CAN CODE
NUL	00			M.P.	12-0-9-8-1	NUL	80	NUL	00		
SOH	01			M.P.	12-9-1	SOM	81	SOH	01		
STX	02			M.P.	12-9-2	EOA	82	STX	02		
ETX	03			M.P.	12-9-3	EOM	83	ETX	03		
EOT	04			M.P.	9-7	EDT	84	EOT	37		
ENQ	05			M.P.	0-9-8-5	WRU	85	ENQ	2D		
ACK	06			M.P.	0-9-8-6	RU	86	ACK	2E		
BEL	07			M.P.	0-9-8-7	BEL	87	BEL	2F		
BS	08			M.P.	11-9-6	FE_O	88	BS	16		
HT	09			M.P.	12-9-5	HT	89	HT	05		
LF	0A			M.P.	0-9-5	LF	8A	LF	25		
VT	0B			M.P.	12-9-8-3	VT	8B	VT	0B		
FF	0C			M.P.	12-9-8-4	FORM	8C	FF	0C		
CR	0D			M.P.	12-9-8-5	RETURN	8D	CR	0D		
S0	0E			M.P.	12-9-8-6	S0	8E	S0	0E		
SI	0F			M.P.	12-9-8-7	SI	8F	SI	0F		
DLE	10			M.P.	12-11-9-8-1	DC0	90	DLE	10		
DC1	11			M.P.	11-9-1	X-ON	91	DC1	11		
DC2	12			M.P.	11-9-2	TAPE	92	DC2	12		
DC3	13			M.P.	11-9-3	X-OFF	93	DC3	13		
DC4	14			M.P.	9-8-4	~~TAPE~~	94	DC4	3C		
NAK	15			M.P.	9-8-5	ERROR	95	NAK	3D		
SYN	16			M.P.	9-2	SYC	96	SYN	32		
ETB	17			M.P.	0-9-6	LEM	97	ETB	26		
CAN	18			M.P.	11-9-8	S0	98	CAN	18		

M.P. = Multi-punch

CHAR.	CODE	L.P.	CODE	CARD READER	CODE 029	TTY	CODE	EBCDIC	CODE	COMP. CHAR.	CAN CODE
EM	19			M.P.	11-9-8-1	S1	99	EM	19		
SUB	1A			M.P.	9-8-7	S2	9A	SUB	3F		
ESC	1B			M.P.	0-9-7	S3	9B	ESC	27		
FS	1C			M.P.	11-9-8-4	S4	9C	FS	1C		
GS	1D			M.P.	11-9-8-5	S5	9D	GS	1D		
RS	1E			M.P.	11-9-8-6	S6	9E	RS	1E		
US	1F			M.P.	11-9-8-7	S7	9F	US	1F		
SPACE	20	SPACE	20	SPACE BAR	-	SPACE	A0	SPACE	40	SPACE	0
!	21	!	21	!	12-8-7	!	A1	!	4F		
"	22	"	22	"	8-7	"	A2	"	7F		
#	23	#	23	#	8-3	#	A3	#	7B		
$	24	$	24	$	11-8-3	$	A4	$	5B	$	39
%	25	%	25	%	0-8-4	%	A5	%	6C		
&	26	&	26	&	12	&	A6	&	50		
'	27	'	27	'	8-5	'	A7	'	7D		
(	28	(	28	(	12-8-5	(	A8	(	4D		
)	29	)	29	)	11-8-5	)	A9	)	5D		
*	2A	*	2A	*	11-8-4	*	AA	*	5C		
+	2B	+	2B	+	12-8-6	+	AB	+	4E		
,	2C	,	2C	,	0-8-3	,	AC	,	6B		
-	2D	-	2D	-	11	-	AD	-	60		
.	2E	.	2E	.	12-8-3	.	AE	.	4B	.	38
/	2F	/	2F	/	0-1	/	AF	/	61		
0	30	0	30	0	0	0	B0	0	F0	0	27
1	31	1	31	1	1	1	B1	1	F1	1	28
2	32	2	32	2	2	2	B2	2	F2	2	29
3	33	3	33	3	3	3	B3	3	F3	3	30
4	34	4	34	4	4	4	B4	4	F4	4	31
5	35	5	35	5	5	5	B5	5	F5	5	32
6	36	6	36	6	6	6	B6	6	F6	6	33

M.P. = Multi-punch

CHAR.	CODE	L.P.	CODE	CARD READER	CODE 029	TTY	CODE	EBCDIC	CODE	COMP. CHAR.	CAN CODE
7	37	7	37	7	7	7	B7	7	F7	7	34
8	38	8	38	8	8	8	B8	8	F8	8	35
9	39	9	39	9	9	9	B9	9	F9	9	36
:	3A	:	3A	:	8-2	:	BA	:	7A	:	37
;	3B	;	3B	;	11-8-6	;	BB	;	5E		
<	3C	<	3C	<	12-8-4	<	BC	<	4C		
=	3D	=	3D	=	8-6	=	BD	=	7E		
>	3E	>	3E	>	0-8-6	>	BE	>	6E		
?	3F	?	3F	?	0-8-7	?	BF	?	6F		
@	40	@	40	@	8-4	@	C0	@	7C		
A	41	A	41	A	12-1	A	C1	A	C1	A	1
B	42	B	42	B	12-2	B	C2	B	C2	B	2
C	43	C	43	C	12-3	C	C3	C	C3	C	3
D	44	D	44	D	12-4	D	C4	D	C4	D	4
E	45	E	45	E	12-5	E	C5	E	C5	E	5
F	46	F	46	F	12-6	F	C6	F	C6	F	6
G	47	G	47	G	12-7	G	C7	G	C7	G	7
H	48	H	48	H	12-8	H	C8	H	C8	H	8
I	49	I	49	I	12-9	I	C9	I	C9	I	9
J	4A	J	4A	J	11-1	J	CA	J	D1	J	10
K	4B	K	4B	K	11-2	K	CB	K	D2	K	11
L	4C	L	4C	L	11-3	L	CC	L	D3	L	12
M	4D	M	4D	M	11-4	M	CD	M	D4	M	13
N	4E	N	4E	N	11-5	N	CE	N	D5	N	14
O	4F	O	4F	O	11-6	O	CF	O	D6	O	15
P	50	P	50	P	11-7	P	D0	P	D7	P	16
Q	51	Q	51	Q	11-8	Q	D1	Q	D8	Q	17
R	52	R	52	R	11-9	R	D2	R	D9	R	18
S	53	S	53	S	0-2	S	D3	S	E2	S	19
T	54	T	54	T	0-3	T	D4	T	E3	T	20

CHAR.	CODE	L.P.	CODE	CARD READER	CODE 029	TTY	CODE	EBCDIC	CODE	COMP. CHAR.	CAN CODE
U	55	U	55	U	0-4	U	D5	U	E4	U	21
V	56	V	56	V	0-5	V	D6	V	E5	V	22
W	57	W	57	W	0-6	W	D7	W	E6	W	23
X	58	X	58	X	0-7	X	D8	X	E7	X	24
Y	59	Y	59	Y	0-8	Y	D9	Y	E8	Y	25
Z	5A	Z	5A	Z	0-9	Z	DA	Z	E9	Z	26
[	5B	[	5B	¢	12-8-2	[	DB	¢	4A		
\	5C	\	5C	0-8-2	0-8-2	\	DC	\	E0		
]	5D	]	5D	!	11-8-2	]	DD	!	5A		
∧	5E	∧	5E	¬	11-8-7	↑	DE	∧	5F		
—	5F	—	5F	—	0-8-5	←	DF	—	6D		
`	60			M.P.	8-1			`	79		
a	61			M.P.	12-0-1			a	81		
b	62			M.P.	12-0-2			b	82		
c	63			M.P.	12-0-3			c	83		
d	64			M.P.	12-0-4			d	84		
e	65			M.P.	12-0-5			e	85		
f	66			M.P.	12-0-6			f	86		
g	67			M.P.	12-0-7			g	87		
h	68			M.P.	12-0-8			h	88		
i	69			M.P.	12-0-9			i	89		
j	6A			M.P.	12-11-1			j	91		
k	6B			M.P.	12-11-2			k	92		
l	6C			M.P.	12-11-3			l	93		
m	6D			M.P.	12-11-4			m	94		
n	6E			M.P.	12-11-5			n	95		
o	6F			M.P.	12-11-6			o	96		
p	70			M.P.	12-11-7			p	97		
q	71			M.P.	12-11-8			q	98		
r	72			M.P.	12-11-9			r	99		

M.P. = Multi-punch

CHAR.	CODE	L.P.	CODE	CARD READER	CODE 029	TTY	CODE	EBCDIC	CODE	COMP. CHAR.	CAN CODE
s	73			M.P.	11-0-2			s	A2		
t	74			M.P.	11-0-3			t	A3		
u	75			M.P.	11-0-4			u	A4		
v	76			M.P.	11-0-5			v	A5		
w	77			M.P.	11-0-6			w	A6		
x	78			M.P.	11-0-7			x	A7		
y	79			M.P.	11-0-8			y	A8		
z	7A			M.P.	11-0-9			z	A9		
{	7B			M.P.	12-0			{	C0		
¦	7C			M.P.	12-11				6A		
}	7D			M.P.	11-0			}	D0		
~	7E			M.P.	11-0-1			~	A1		
DEL	7F			M.P.	12-9-7	RUB OUT	FF	DEL	07		

APPENDIX C

MOVING HEAD DISC (CONTROLLER ADDRESS 01_{16})

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE	WRITE	1	M	CONN. DI	CONN. SI	0	EOD	EOF	IGN	SCS	IGNORED		S	S	S	S	S
	READ	1	M	CONN. DI	CONN. SI	1	IGNORED			SCS	IGNORED		S	S	S	S	S
CONTROL	CYLINDER SELECT	0	1	CONN. DI	CONN. SI	0	0	1	*C	C	C	C	C	C	C	C	C
	TERM/ EOB	0	1	IGNORED		EOB	TERM.	IGN	MPE	IGNORED							
	HEAD/ DRIVE SELECT	0	0	IGN	*HEAD SEL	*HEAD SEL	*HEAD SEL	*HEAD SEL	HEAD SEL	CONTIN-UOUS SCAN	IGNORED				1= PREP MODE	U	U
	NO-OP	0	1	CONN. DI	CONN. SI	0	0	0	IGNORED								
	STATUS	0= ERROR	O/F U/F	CRC	IN OP	MPE	WLO	SEEK ERROR	1= BUSY	0= DATA READY	EOD	EOF	EOR	DEV. SEEK	SEEK COMP.	U	U

S = SECTOR, C = CYLINDER, DMP LOCATIONS: TC = 61, TA = 71

U = UNIT NO., UP TO 4 UNITS MAY BE CONNECTED TO A CONTROLLER

M = MODE, IF BIT 1 = 0 PROGRAMMED I/O IS SELECTED RATHER THAN DMP

* THESE BITS ARE USED ONLY IN DISC PACK MOVING HEAD DISC CONTROLLERS.

FIXED HEAD DISC (CONTROLLER ADDRESS 02_{16})

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE	WRITE	1	M	CONN. DI	CONN. SI	0	EOD	EOF	IGNORED				S	S	S	S	S
	READ	1	M	CONN. DI	CONN. SI	1	IGNORED						S	S	S	S	S
CONTROL	HEAD SELECT	0	1	CONN. DI	CONN. SI	0	0	1	IGN	H	H	H	H	H	H	H	H
	TERM/ EOB	0	1	IGNORED		EOB	TERM.	IGN.	MPE	IGNORED							
	UNIT* SELECT	0	0	IGNORED												U	U
	NO-OP	0	1	CONN. DI	CONN. SI	0	0	ø		IGNORED							
	STATUS	0= ERROR	O/F U/F	CK SUM	IN OP	MPE	WLO	ø	1= BUSY	0= DATA READY	EOD	EOF	EOR	0	0	0	0

S = SECTOR, H = HEAD, DMP LOCATIONS: TC = 62, TA = 72

U = UNIT NO., UP TO 4 UNITS MAY BE CONNECTED TO A CONTROLLER

M = MODE, IF BIT 1 = 0 PROGRAMMED I/O IS SELECTED RATHER THAN DMP

* THIS COMMAND NOT USED WITH MODEL 4103, 4104, 4105.

MAGNETIC TAPE (DEVICE ADDRESS, HIGH SPEED 03_{16}, LOW SPEED 04_{16})

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE	WRITE	1	M	CONN. DI	CONN. SI	0	0= BINARY 1= INTER-CHANGE	0= N-GAP 1= L-GAP	1= SINGLE CYCLE SCAN	0= ODD 1= EVEN	0= 800 CPI 1= 556 CPI	0	0	0	0	U	U
	READ	1	M	CONN. DI	CONN. SI	1	0= BINARY 1= INTER-CHANGE	0	1=SCS	0= ODD 1= EVEN	0= 800 CPI 1= 556 CPI	0	0	0	0	U	U
CONTROL	WRITE EOF	0	1	CONN. DI	CONN. SI	0	0	1	1=SCS	0	0= 800 CPI 1= 556 CPI	WRITE EOF 1	0	0	0	U	U
	SPACE	0	1	CONN. DI	CONN. SI	0	0	1	1=SCS	0	0= 800 CPI 1= 556 CPI	0	SPACE 1	0= FORWARD 1= REVERSE	0= BLOCK 1= FILE	U	U
	EOB/ TERM	0	1	IGN	IGN	1=EOB	1=TERM	IGN	1=MPE	IGNORED							
	REWIND	0	1	CONN. DI	CONN. SI	0	0	1	1= LOCK OUT	REWIND 1	0	0	0	0	0	U	U
	TRANSPORT SELECT	0	1	CONN. DI	CONN. SI	0	0	1	1= CONT. SCAN	0	0	0	0	0	0	U	U
	NO-OP	0	1	CONN. DI	CONN. SI	0	0	0	IGNORED								
STATUS		0= ERROR	1= OVER/ FLOW OR B>C	1= DEVICE PARITY ERROR	1= IN OP	1= MEMORY PARITY ERROR	1= FILE PROTECT	1= TAPE DETECT	1= CONT. BUSY	0= DATA READY	1= EOT	1= EOF	1= BOT	1= DEVICE OFFLINE OR REWIND	1= PARTIAL WORD	U	U

U = UNIT NO., UP TO 4 UNITS MAY BE CONNECTED TO A CONTROLLER
M = MODE, IF BIT 1 = 0 PROGRAMMED I/O IS SELECTED RATHER THAN DMP
DMP LOCATIONS (HEX.) TC=63, TA=73 TC=64, TA=74

CARD READER (DEVICE ADDRESS 05_{16})

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE		1	0	CONN. DI	CONN. SI	1	1=BIN 0=XLATE	IGNORED									
CONTROL	EOB/ TERM	0	1	IGNORED		EOB	TERM.	IGNORED									
	NO-OP	0	1	CONN. DI	CONN. SI	0	0	0	IGNORED								
STATUS		0= ERROR	1= OVER-FLOW	1= READ CHECK ERROR	1= IN OP	0	0	0	1= BUSY	0= DATA BUFFER READY	1= HOPPER EMPTY/ STACKER FULL	0	0	1= HOLD	1= PICK FAIL	0	0

CARD PUNCH (DEVICE ADDRESS 06_{16})

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE		1	0	CONN. DI	CONN. SI	0	A SEE TABLE*	B	1= OFFSET STACK	IGNORED							
CONTROL	EOB/ TERM	0	1	IGNORED		EOB	TERM.	IGNORED									
	NO-OP	0	1	CONN. DI	CONN SI	0	0	0	IGNORED								
STATUS		0= ERROR	1= UNDER-FLOW	1= PUNCH ERROR	1= DEV. IN.OP	0	0	0	1= CONT. BUSY	0= BUFFER EMPTY	1= HOPPER EMPTY/ STACKER FULL	0	0	0	1= TRANS-PORT	0	0

*AB	OUTPUT TRANSLATION MODES
11	ILLEGAL
01	XLATE, FROM ASCII (7 BIT) TO HOLLERITH
10	12 BIT BINARY (1 TO 1)
00	8 BIT BINARY EXPANDED TO 12 COL. PUNCH

LINE PRINTER 600 LPM (DEVICE ADDRESS 07_{16})

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE		1	0	CONN. DI	CONN. SI	0	IGNORED				1=VFU 0=L.CNT	LINE COUNT	LINE COUNT	LINE COUNT OR VFU	LINE COUNT OR VFU	LINE COUNT OR VFU	LINE COUNT OR VFU
CONTROL	EOB/ TERM	0	1	IGNORED		EOB	TERM.	1=LINE FEED	IGN		1=VFU 0=L.CNT	LINE COUNT	LINE COUNT	LINE COUNT OR VFU	LINE COUNT OR VFU	LINE COUNT OR VFU	LINE COUNT OR VFU
	NO-OP	0	1	CONN. DI	CONN. SI	0	0	D	IGNORED								
STATUS		0= ERROR	0	0	1= IN OP	0	0	0	1= BUSY	0= DATA BUFFER READY	1= PAPER LOW	1= BOTTOM OF FORM	0	1= HOLD	0	0	0

ELECTROSTATIC PRINTER/PLOTTER (DEVICE ADDRESS 08_{16})

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE		1	M	CONN. DI	CONN. SI	0	*PLOT	**SPP	← IGNORED →								
CONTROL	EOB/ TERM	0	1	IGNORED		EOB	TERM	IGN	MPE	← IGNORED →							
	NO-OP	0	1	CONN. DI	CONN. SI	0	0	0	← IGNORED →								
STATUS		0= ERROR	0	0	INOP	MPE	0	0	1= BUSY	0= DATA READY	PAPER LOW	0	0	0	0	0	0

*PLOT: 1 = PLOT
0 = PRINT

**SPP: 1 = SIMULTANEOUS PRINT/PLOT
0 = NORMAL

X-Y PLOTTER (DEVICE ADDRESS 08_{16})

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE		1	0	CONN. DI	CONN. SI	0	IGNORED										
CONTROL	EOB/ TERM	0	1	IGNORED		1= EOB	1= TERM	IGNORED									
	OUTPUT DATA	IGNORED										1=PEN DOWN	1=PEN UP	1=DRUM DOWN	1=DRUM UP	1=CARR. RIGHT	1=CARR. LEFT
												**		**		**	
	NO-OP	0	1	CONN. DI	CONN. SI	0	0	0	IGNORED								
STATUS		0= ERROR	0	0	1= IN OP	0	0	0	1= BUSY	0= DATA	0	0	0	0	0	0	0

**MUTUALLY EXCLUSIVE

PAPER TAPE PUNCH (DEVICE ADDRESS 09_{16})

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE		1	0	CONN. DI	CONN. SI	0	IGNORED										
CONTROL	EOB/TERM	0	1	IGNORED		EOB	TERM.	IGN	1= ABORT	IGNORED							
	NO-OP	0	1	CONN. DI	CONN. SI	0	0	0	IGNORED								
STATUS		1	0	0	0	0	0	0	1= BUSY	0= DATA READY	1= TAPE LOW	0	0	0	0	0	0

CONSOLE TTY/PAPER TAPE READER (DEVICE ADDRESS $0A_{16}$)

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE		1	0	CONN. DI	CONN. SI	1=IN 0=OUT	1= KEY BD.	ENABLE CLOCK	CLEAR BUFFER	IGNORED				0=REV.* 1=FOR.	IGNORED		
CONTROL	EOB/TERM	0	1	IGNORED		0	TERM	IGN	1=ABORT	IGNORED							
	NO-OP	0	1	CONN. DI	CONN. SI	0	0	0	IGNORED								
STATUS		1	0	0	0	0	0	0	1= BUSY	0= DATA READY	0	0	0	0	0	0	0

*NOTE: HIGH SPEED READER NORMALLY OPERATES IN REVERSE MODE.

LINE PRINTER 50-150 LPM (DEVICE ADDRESS $0B_{16}$)

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE		1	0	CONN. DI	CONN. SI	0	IGNORED										
CONTROL	EOB/TERM	0	1	IGNORED		EOB	TERM	IGNORED									
	NO-OP	0	1	CONN. DI	CONN. SI	0	0	0	IGNORED								
STATUS		0= ERROR	0	0	1= IN OP	0	0	0	1= CONT. BUSY	0= DATA READY	1= LOW PAPER	1= BOTTOM OF FORM	0	1= HOLD	0	0	0

HIGH LEVEL AIS OUTPUT UNIT (DEVICE ADDRESS 10_{16})

<table>
<tr><th colspan="2"></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td colspan="2">TRANSFER INITIATE</td><td>1</td><td>M</td><td>CONN. DI</td><td>IGN</td><td>1=SEQ
0=RAN</td><td colspan="4">IGNORED</td><td colspan="7">*END ADDRESS</td></tr>
<tr><td rowspan="2">CONTROL</td><td>EOB/ TERM</td><td>0</td><td>1</td><td colspan="2">IGNORED</td><td>EOB</td><td>TERM</td><td colspan="3">IGNORED</td><td colspan="7">*START ADDRESS</td></tr>
<tr><td>NO-OP</td><td>0</td><td>1</td><td>CONN. DI</td><td>IGN</td><td>0</td><td>0</td><td>0</td><td colspan="9">IGNORED</td></tr>
<tr><td colspan="2">STATUS</td><td>ERROR</td><td colspan="6">IGNORED</td><td>1=
BUSY</td><td>0=
DATA READY</td><td colspan="7">IGNORED</td></tr>
<tr><td colspan="2">DATA WORD</td><td colspan="9">IGNORED</td><td colspan="7">* CHANNEL ADDRESS</td></tr>
</table>

* ADDRESSES ARE IN BINARY (0-127)

HIGH LEVEL AIS INPUT UNIT (DEVICE ADDRESS 11_{16})

<table>
<tr><th colspan="2"></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td colspan="2">TRANSFER INITIATE</td><td>1</td><td>M</td><td>CONN. DI</td><td>CONN. SI</td><td colspan="12">IGNORED</td></tr>
<tr><td rowspan="2">CONTROL</td><td>EOB/ TERM</td><td>0</td><td>1</td><td colspan="2">IGNORED</td><td>EOB</td><td>TERM</td><td colspan="10">IGNORED</td></tr>
<tr><td>NO-OP</td><td>0</td><td>1</td><td>CONN. DI</td><td>CONN. SI</td><td>0</td><td>0</td><td>0</td><td colspan="9">IGNORED</td></tr>
<tr><td colspan="2">STATUS</td><td>0=
ERROR</td><td colspan="6">IGNORED</td><td>1=
BUSY</td><td>0=
DATA READY</td><td colspan="7">IGNORED</td></tr>
<tr><td colspan="2" rowspan="2">DATA WORD</td><td rowspan="2">SIGN</td><td rowspan="2">1</td><td rowspan="2">0</td><td rowspan="2">1</td><td rowspan="2">1</td><td colspan="11">DATA</td></tr>
<tr><td>2^{10}</td><td>2^9</td><td>2^8</td><td>2^7</td><td>2^6</td><td>2^5</td><td>2^4</td><td>2^3</td><td>2^2</td><td>2^1</td><td>2^0</td></tr>
</table>

WIDE RANGE SOLID STATE AIS OUTPUT UNIT (DEVICE ADDRESS 12_{16})

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE		1	M	CONN. DI	IGNORED		1= RE-CYCLE	← IGNORED →									
CONTROL	EOB/TERM	0	1	IGNORED		EOB	TERM	← IGNORED →									
	NO-OP	0	1	CONN. DI	IGN	0	0	0	← IGNORED →								
STATUS		ERROR	IGNORED						1= BUSY	0= DATA READY	IGNORED						
DATA WORD 1		IGN	GAIN 2^3	2^2	2^1	2^0	1= ZERO SUPPR.	1= AUTO	IGNORED		* CHANNEL ADDRESS $(0-127)_2$ 2^6	2^5	2^4	2^3	2^2	2^1	2^0
SUPPRESSION WORD		SIGN	IGN	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

* ADDRESSES ARE IN BINARY (0-127)

WIDE RANGE SOLID STATE AIS INPUT UNIT (DEVICE ADDRESS 13_{16})

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE		1	M	CONN. DI	CONN. SI	← IGNORED →											
CONTROL	EOB/TERM	0	1	IGNORED		EOB	TERM	← IGNORED →									
	NO-OP	0	1	CONN. DI	CONN. SI	0	0	0	← IGNORED →								
STATUS		ERROR	IGNORED						BUSY	DATA READY	IGNORED						
DATA WORD		SIGN	GAIN 2^3	2^2	2^1	2^0	← DATA → 2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

WIDE RANGE RELAY AIS (DEVICE ADDRESS 14_{16})

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE		1	0	CONN. DI	CONN. SI	IGNORED											
CONTROL	EOB/ TERM	0	1	IGNORED		EOB	TERM	IGNORED									
	NO-OP	0	1	CONN. DI	CONN. SI	0	0	0	IGNORED								
STATUS		ERROR	IGNORED						BUSY	INPUT DATA READY	IGNORED						OUTPUT DATA READY
DATA WORD 1		1=RES. CHECK MODE	GAIN				1= ZERO SUPPR.	1= AUTO	* CHANNEL ADDRESS								
SUPPRESSION WORD		SIGN	IGN.	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

* ADDRESSES ARE IN BINARY (0-511)

MODAC SUBSYSTEM 1 (DEVICE ADD. 20_{16}-$2B_{16}$)
SUBSYSTEM 2 (DEVICE ADD. 30_{16}-$3B_{16}$)

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE	AIM	1	M	CONN. DI	CONN. SI	1= SEQ	IGN	END ADDRESS 2^4	2^3	2^2	2^1	2^0	START ADDRESS 2^4	2^3	2^2	2^1	2^0
CONTROL	EOB/TERM AIM	0	1	IGNORED		EOB	TERM	IGNORED									
	NO-OP AIM	0	1	CONN. DI	CONN. SI	0	0	0	IGNORED								
	INTERRUPT EXTENDER	0	1	CONN. DI	CONN. SI	IGNORED			R	INTERRUPT CHANNEL 0	1	2	3	4	5	6	7
STATUS	AIM	ERROR	IGNORED						BUSY	0=IN DATA RDY	IGNORED						0=OUT DATA RDY
	SYNCHRONIZER	ERROR	IGNORED							1=XFER REQ	IGNORED						
	INTERRUPT EXTENDER	DATA INTERRUPT 0	1	2	3	4	5	6	7	SERVICE INTERRUPT 0	1	2	3	4	5	6	7
DATA	OUTPUT AIM	IGNORED											CHANNEL 2^4	2^3	2^2	2^1	2^0
	INPUT AIM	SIGN					DATA 2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	ANALOG OUTPUT (VOLTAGE)	SIGN	ADDRESS 2^1	2^0	CRT CONT.	CRT CONT.	DATA 2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	ANALOG OUTPUT (CURRENT)	IGN	ADDRESS 2^1	2^0	IGN	DATA 2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

I/OIS#1 (DEVICE ADDRESS 20-2F$_{16}$) I/OIS#2 (DEVICE ADDRESS 30-3F$_{16}$)

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CONTROL	I/O INTERRUPT COUPLER	0	1	CONN. DI	CONN. SI	IGNORED			1=RES REQ.	INTERRUPT CHANNEL 0	1	2	3	4	5	6	7
	INTERVAL TIMER	0	1	IGNORED							0=SINGLE 1=RECYCLE	IGNORED					
	CHANNEL MULTIPLEXER	0	1	IGNORED								EXPANDER ADDRESS			CHANNEL ADDRESS		
	SACI	0	1	IGNORED		0	T	IGNORED									
	SACI NO-OP	0	1	CONN. DI	CONN. SI	0	0	0	IGNORED								
TRANSFER INITIATE	SACI	1	0	CONN. DI	CONN. SI	1= IN	1= KEY BD.	IGNORED									
STATUS	SACI	ERROR	IGNORED		1=DATA SET NOT READY	IGNORED			BUSY	DATA READY	IGNORED						
	SYNCHRONIZER	SYS. CONN.	IGNORED							1= XFER REQ.	IGNORED						
	SUBSYSTEM	SYS CONN.	IGNORED														
DATA	ANALOG OUTPUT	SIGN	0=CH1 1=CH2	CRT CONT.	CRT CONT.	CRT CONT.	DATA 2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	SACI	IGNORED								ASCII DATA							

APPENDIX D. DIVIDE

During execution of a divide instruction, the contents of registers Ra, RaV1 (where a is an even number $\neq 0$) form a double precision dividend and are divided by the single-precision divisor specified by the instruction. If the dividend is a single-precision number, RaV1 should be cleared prior to executing the divide instruction or erroneous results may occur. Although a double-length dividend is used, divide is a single-precision operation and should not be confused with a double-precision divide operation that would use a double-length divisor and would produce a double-length quotient.

After execution of the divide, the single-precision quotient replaces the contents of RaV1 and the remaining portion of the dividend that has not been divided (undivided remainder) replaces the contents of Ra. The quotient is signed in accordance with algebraic convention, that is, positive if dividend and divisor signs are alike, but negative otherwise. However, only 15 magnitude bits are generated by the divide and, if the magnitude of the quotient is so small as to require more than 15 bits to resolve, a zero quotient may be generated regardless of the required sign, but the remainder will still reflect the undivided portion of the original dividend. The binary scaling of the quotient is equal to the dividend scale factor minus the divisor scale factor.

The undivided remainder replaces the contents of Ra and has the same sign as the dividend. It has the same scaling as the divisor. By definition, the undivided remainder is that quantity which must be added to the product of the divisor and the quotient to produce the original dividend. The results of the divide instruction are consistent with this definition. It should be noted that the remainder must be added to the least significant part of the product of the divisor and the quotient to maintain proper scaling. Overflow is possible and the Overflow Indicator will be set if:

$$\left| \frac{(Ra,\ RaV1)}{(M)} \right| \geq 1$$

Reference section on Overflow (Page 3-9)

EXAMPLE: Let (Ra, RaV1) = 0004 E800
(M) = 3C00

The dividend can be represented as a decimal 628 with an equivalent binary point at 2^{21}. The divisor may be represented as a decimal 30 with its binary point at 2^{6}. The resulting binary scaling of the quotient is $2^{21}-2^{6}=2^{15}$. The remainder is scaled at 2^{6}.

Q = 0014 (20_{10} at 2^{15}) R = 3800 (28_{10} at 2^{6})

APPENDIX E. INSTRUCTION LIST

MNEMONIC	OP. CODE	NAME	EXECTION TIME (μs)	PAGE
LOAD, STORE AND TRANSFER				
LDM	E5	Load Register from Memory	2.4	3-4
LDI	ED	Load Register from Memory Immediate	1.6	3-4
LDS	F5	Load Register from Memory Short Displaced	1.87	3-4
LDX	FD	Load Register from Memory Short Indexed	1.87	3-5
STM	E6	Store Register in Memory	2.4	3-5
STI	EE	Store Register in Memory Immediate	1.6	3-5
STS	F6	Store Register in Memory Short Displaced	1.87	3-5
STX	FE	Store Register in Memory Short Indexed	1.87	3-6
LBX	AE	Load Byte From Memory	2.13	3-6
SBX	AF	Store Byte in Memory	3.2	3-7
LFM	A4	Load File from Memory	4.0 + .8 (R-1)	3-7
LFS	B4	Load File from Memory Short Displaced	3.47 + .8 (R-1)	3-7
LFX	BC	Load File from Memory Short Indexed	3.47 + .8 (R-1)	3-8
SFM	A5	Store File in Memory	4.0 + .8 (R-1)	3-8
SFS	B5	Store File in Memory Short Displaced	3.47 + .8 (R-1)	3-8
SFX	BD	Store File in Memory Short Indexed	3.47 + .8 (R-1)	3-8
TRR	6D	Transfer Register to Register	0.8	3-9
TRRB	7D	Transfer Register to Register and Branch if Nonzero	1.6	3-9

ARITHMETIC

MNEMONIC	OP. CODE	NAME	EXECTION TIME (μs)	PAGE
ADM	E0	Add Memory to Register	2.4	3-11
ADI	E8	Add Memory to Register Immediate	1.6	3-11
ADS	F0	Add Memory to Register Short Displaced	1.87	3-11
ADX	F8	Add Memory to Register Short Indexed	1.87	3-12
ADMM	C0	Add Register to Memory	3.47	3-12
ADMB	C4	Add Register to Memory and Branch if Nonzero	4.27	3-12
ADSM	D0	Add Register to Memory Short Displaced	2.67	3-13
ADSB	D4	Add Register to Memory Short Displaced and Branch if Nonzero	3.47	3-13
ADXM	D8	Add Register to Memory Short Indexed	2.67	3-13
ADXB	DC	Add Register to Memory Short Indexed and Branch if Nonzero	3.47	3-14
ADR	68	Add Register to Register	0.8	3-14
ADRB	78	Add Register to Register and Branch if Nonzero	1.6	3-14

MNEMONIC	OP. CODE	NAME	EXECUTION TIME (us)	PAGE
ARITHMETIC (CONTINUED)				
DAR	22	Double Precision Add Register to Register	2.13	3-15
SUM	E1	Subtract Memory from Register	2.4	3-15
SUI	E9	Subtract Memory from Register Immediate	1.6	3-15
SUS	F1	Subtract Memory from Register Short Displaced	1.87	3-16
SUX	F9	Subtract Memory from Register Short Indexed	1.87	3-16
SUR	69	Subtract Register from Register	0.8	3-16
SURB	79	Subtract Register from Register and Branch if Nonzero	1.6	3-16
MPM	A0	Multiply Memory by Register	7.74	3-17
MPS	B0	Multiply Memory by Register Short Displaced	7.21	3-17
MPX	B8	Multiply Memory by Register Short Indexed	7.21	3-17
MPR	20	Multiply Register by Register	6.67	3-18
DVM	A1	Divide Register by Memory	12.2	3-18
DVS	B1	Divide Register by Memory Short Displaced	11.4	3-18
DVX	B9	Divide Register by Memory Short Indexed	11.4	3-19
DVR	21	Divide Register by Register	11.0	3-19
CRMB	C7	Compare Memory and Register	4.53	3-19
CRSB	D7	Compare Memory and Register Short Displaced	4.0	3-20
CRXB	DF	Compare Memory and Register Short Indexed	4.0	3-20
TRO	0E	Transfer and Reset Overflow Status	0.8	3-20
TTR	6F	Transfer Two's Complement Register to Register	0.8	3-21
TTRB	7F	Transfer Two's Complement Register to Register and Branch if Nonzero	1.6	3-21
LOGICAL				
ETM	E2	Extract Memory from Register	2.4	3-22
ETI	EA	Extract Memory from Register Immediate	1.6	3-22
ETS	F2	Extract Memory from Register Short Displaced	1.87	3-22
ETX	FA	Extract Memory from Register Short Indexed	1.87	3-23
ETMM	C1	Extract Register from Memory	3.47	3-23
ETMB	C5	Extract Register from Memory and Branch if Nonzero	4.27	3-23
ETSM	D1	Extract Register from Memory Short Displaced	2.67	3-24
ETSB	D5	Extract Register from Memory Short Displaced and Branch if Nonzero	3.47	3-24
ETXM	D9	Extract Register from Memory Short Indexed	2.67	3-24
ETXB	DD	Extract Register from Memory Short Indexed and Branch if Nonzero	3.47	3-25
ETR	6A	Extract Register from Register	0.8	3-25
ETRB	7A	Extract Register from Register and Branch if Nonzero	1.6	3-25
ORM	E3	OR Memory and Register	2.4	3-26
ORI	EB	OR Memory and Register Immediate	1.6	3-26
ORS	F3	OR Memory and Register Short Displaced	1.87	3-26
ORX	FB	OR Memory and Register Short Indexed	1.87	3-26

MNEMONIC	OP. CODE	NAME	EXECUTION TIME (us)	PAGE
LOGICAL (CONTINUED)				
ORMM	C2	OR Register and Memory	3.47	3-27
ORSM	D2	OR Register and Memory Short Displaced	2.67	3-27
ORXM	DA	OR Register and Memory Short Indexed	2.67	3-27
ORR	6B	OR Register and Register	0.8	3-27
ORRB	7B	OR Register and Register and Branch if Nonzero	1.6	3-28
XOM	E4	Exclusive OR Memory and Register	2.4	3-28
XOI	EC	Exclusive OR Memory and Register Immediate	1.6	3-28
XOS	F4	Exclusive OR Memory and Register Short Displaced	1.87	3-29
XOX	FC	Exclusive OR Memory and Register Short Indexed	1.87	3-29
XOR	6C	Exclusive OR Register and Register	0.8	3-29
XORB	7C	Exclusive OR Register and Register and Branch if Nonzero	1.6	3-29
TOR	0D	Transfer One's Complement Register to Register	0.8	3-30
TRMB	C6	Test Register and Memory and Branch if Any Ones Compare	3.73	3-30
TRSB	D6	Test Register and Memory Short Displaced and Branch if Any Ones Compare	2.99	3-30
TRXB	DE	Test Register and Memory Short Indexed and Branch if Any Ones Compare	2.93	3-31
TERB	7E	Test Register and Register and Branch if Any Ones Compare	1.6	3-31
FLOATING POINT				
FAR	30	Floating Add Register to Register	15.0	3-34
FSR	31	Floating Subtract Register from Register	15.0	3-35
FMR	32	Floating Multiply Register by Register	12.5	3-35
FDR	33	Floating Divide Register by Register	13.0	3-35
FARD	34	Floating Add Register to Register Double	20.5	3-36
FSRD	35	Floating Subtract Register from Register Double	20.5	3-36
FMRD	36	Floating Multiply Register by Register Double	16.0	3-37
FDRD	37	Floating Divide Register by Register Double	16.5	3-37
FAM	38	Floating Add Memory to Register	17.5	3-37
FSM	39	Floating Subtract Memory from Register	17.5	3-38
FMM	3A	Floating Multiply Memory by Register	14.5	3-38
FDM	3B	Floating Divide Memory into Register	15.5	3-39
FAMD	3C	Floating Add Memory to Register Double	22.5	3-39
FSMD	3D	Floating Subtract Memory from Register Double	22.5	3-39
FMMD	3E	Floating Multiply Memory by Register Double	18.0	3-40
FDMD	3F	Floating Divide Memory into Register Double	19.0	3-40

MNEMONIC	OP. CODE	NAME	EXECUTION TIME (μs)	PAGE
SHIFT				
LAD	2E	Shift Left Arithmetic Double	1.87 + 267(N-1)	3-41
RAD	2A	Shift Right Arithmetic Double	1.87 + 267(N-1)	3-41
LAS	2F	Shift Left Arithmetic Single	1.6 + 267(N-1)	3-42
RAS	2B	Shift Right Arithmetic Single	1.6 + 267(N-1)	3-42
LLD	2C	Shift Left Logical Double	1.867 + 267(N-1)	3-42
RLD	28	Shift Right Logical Double	1.867 + 267(N-1)	3-42
LLS	2D	Shift Left Logical Single	1.6 + 267(N-1)	3-43
RLS	29	Shift Right Logical Single	1.6 + 267(N-1)	3-43
LRS	OF	Left Rotate Single	0.8	3-43
BIT MANIPULATION				
LBR	65	Load Bit in Register	0.8	3-44
LBRB	75	Load Bit in Register and Branch Unconditionally	1.6	3-44
ABMM	80	Add Bit in Memory	3.47	3-45
ABMB	84	Add Bit in Memory and Branch if Nonzero	4.27	3-45
ABSM	90	Add Bit in Memory Short Displaced	2.67	3-45
ABSB	94	Add Bit in Memory Short Displaced and Branch if Nonzero	3.74	3-46
ABXM	98	Add Bit in Memory Short Indexed	2.67	3-46
ABXB	9C	Add Bit in Memory Short Indexed and Branch if Nonzero	3.74	3-46
ABR	60	Add Bit in Register	0.8	3-47
ABRB	70	Add Bit in Register and Branch if Nonzero	1.6	3-47
SBR	61	Subtract Bit in Register	0.8	3-47
SBRB	71	Subtract Bit in Register and Branch if Nonzero	1.6	3-48
ZBMM	81	Zero Bit in Memory	3.47	3-48
ZBMB	85	Zero Bit in Memory and Branch if Nonzero	4.8	3-48
ZBSM	91	Zero Bit in Memory Short Displaced	2.67	3-49
ZBSB	95	Zero Bit in Memory Short Displaced and Branch if Nonzero	4.27	3-49
ZBXM	99	Zero Bit in Memory Short Indexed	2.93	3-49
ZBXB	9D	Zero Bit in Memory Short Indexed and Branch if Nonzero	4.27	3-49
ZBR	62	Zero Bit in Register	0.8	3-50
ZBRB	72	Zero Bit in Register and Branch if Nonzero	1.6	3-50
OBMM	82	OR Bit in Memory	3.47	3-50
OBSM	92	OR Bit in Memory Short Displaced	2.67	3-50
OBXM	9A	OR Bit in Memory Short Indexed	2.94	3-51
OBR	63	OR Bit in Register	0.8	3-51
OBRB	73	OR Bit in Register and Branch Unconditionally	1.6	3-51
XBR	64	Exclusive OR Bit in Register	0.8	3-51
XBRB	74	Exclusive OR Bit in Register and Branch if Nonzero	1.6	3-52
TBMB	86	Test Bit in Memory and Branch if One	3.47	3-52
TBSB	96	Test Bit in Memory Short Displaced and Branch if One	3.2	3-52
TBXB	9E	Test Bit in Memory Short Indexed and Branch if One	3.2	3-53
TBRB	76	Test Bit in Register and Branch if One	1.6	3-53
CBMB	87	Compare Bit and Memory	4.27	3-53
CBSB	97	Compare Bit and Memory Short Displaced	3.7	3-54
CBXB	9F	Compare Bit and Memory Short Indexed	3.7	3-54
GMR	67	Generate Mask in Register	0.8	3-55
GMRB	77	Generate Mask in Register and Branch Unconditionally	1.6	3-55

MNEMONIC	OP. CODE	NAME	EXECUTION TIME (μs)	PAGE
BYTE MANIPULATION				
MUR	0B	Move Upper Byte Register to Register	0.8	3-56
MLR	0C	Move Lower Byte Register to Register	0.8	3-56
MBR	08	Move Byte Right Register to Register	0.8	3-56
MBL	09	Move Byte Left Register to Register	0.8	3-57
IBR	0A	Interchange Bytes Register to Register	0.8	3-57
UNCONDITIONAL BRANCH				
BLM	E7	Branch and Link	1.6	3-58
BLI	EF	Branch and Link Immediate	0.8	3-58
BRU	E7	Branch Unconditionally	1.6	3-58
HOP	F7	Branch Short Displaced	1.07	3-59
BRX	FF	Branch Short Indexed	1.07	3-59
HLT	00	Halt	---	3-60
NOP	66	No Operation	0.8	3-60
CONTROL				
SPR	02-0	Set Protect Register	0.8	3-60
SGP	02-8	Set Global Protect Register	0.8	3-61
SLP	03	Set Lower Protect Register	0.8	3-61
SUP	04	Set Upper Protect Register	0.8	3-61
INTERRUPT AND CALL				
SIE	26-4	Set Interrupt Enable	1.2	3-61
RIE	27-4	Reset Interrupt Enable	1.33	3-61
SIR	26-8	Set Interrupt Request	1.33	3-62
RIR	27-8	Reset Interrupt Request	1.33	3-62
SIA	26-0	Set Interrupt Active	1.33	3-62
RIA	27-0	Reset Interrupt Active	1.33	3-62
REX	23	Request Executive Service	1.07	3-62
RMI	01	Request Multiprocessor Interrupt	0.8	3-63
CAR	24	Clear Active and Return	1.87	3-63
CIR	25	Clear Interrupt and Return	1.87	3-63
INPUT/OUTPUT				
ISA	48	Input Status from I/O Group A	1.6	3-64
ISB	49	Input Status from I/O Group B	1.6	3-64
ISC	4A	Input Status from I/O Group C	1.6	3-64
ISD	4B	Input Status from I/O Group D	1.6	3-64
IDA	4C	Input Data from I/O Group A	1.6	3-65
IDB	4D	Input Data from I/O Group B	1.6	3-65
IDC	4E	Input Data from I/O Group C	1.6	3-65
IDD	4F	Input Data from I/O Group D	1.6	3-65
OCA	40	Output Command to I/O Group A	1.33	3-65
OCB	41	Output Command to I/O Group B	1.33	3-65
OCC	42	Output Command to I/O Group C	1.33	3-65
OCD	43	Output Command to I/O Group D	1.33	3-65
ODA	44	Output Data to I/O Group A	1.33	3-66
ODB	45	Output Data to I/O Group B	1.33	3-66
ODC	46	Output Data to I/O Group C	1.33	3-66
ODD	47	Output Data to I/O Group D	1.33	3-66

APPENDIX F. TABLE OF POWERS OF TWO AND SIXTEEN

16^k 2^n	n	k	2^{-n}
1	0	0	1.0
2	1		0.5
4	2		0.25
8	3		0.125
16	4	1	0.062 5
32	5		0.031 25
64	6		0.015 625
128	7		0.007 812 5
256	8	2	0.003 906 25
512	9		0.001 953 125
1 024	10		0.000 976 562 5
2 048	11		0.000 488 281 25
4 096	12	3	0.000 244 140 625
8 192	13		0.000 122 070 312 5
16 384	14		0.000 061 035 156 25
32 768	15		0.000 030 517 578 125
65 536	16	4	0.000 015 258 789 062 5
131 072	17		0.000 007 629 394 531 25
262 144	18		0.000 003 814 697 265 625
524 288	19		0.000 001 907 348 632 812 5
1 048 576	20	5	0.000 000 953 674 316 406 25
2 097 152	21		0.000 000 476 837 158 203 125
4 194 304	22		0.000 000 238 418 579 101 562 5
8 388 608	23		0.000 000 119 209 289 550 781 25
16 777 216	24	6	0.000 000 059 604 664 775 390 625
33 554 432	25		0.000 000 029 802 322 387 695 312 5
67 108 864	26		0.000 000 014 901 161 193 847 656 25
134 217 728	27		0.000 000 007 450 580 596 923 828 125
268 435 456	28	7	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29		0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30		0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31		0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	8	0.000 000 000 232 830 643 653 869 628 906 25

APPENDIX G.

EXAMPLES OF FLOATING POINT NUMBERS

NUMBER	MACHINE REPRESENTATION
*** SINGLE PRECISION	
16.000	41F00000
15.000	413C0000
14.000	41380000
13.000	41340000
12.000	41300000
11.000	412C0000
10.000	41280000
9.000	41240000
8.000	41200000
7.000	40F80000
6.000	40F00000
5.000	40E80000
4.000	40E00000
3.000	40C00000
2.000	40A00000
1.000	40800000
0.000	0
-1.000	BFA00000
-2.000	BFC00000
-3.000	BF500000
-4.000	BF200000
-5.000	BF180000
-6.000	BF100000
-7.000	BF080000
-8.000	BEE00000
-9.000	BEEC0000
-10.000	BEE80000
-11.000	BEC40000
-12.000	BEB00000
-13.000	BECC0000
-14.000	BEC80000
-15.000	BEC40000
1.000	40800000
0.938	403C0000
0.875	40380000
0.812	40340000
0.750	40300000
0.688	402C0000
0.625	40280000
0.563	40240000
0.500	40200000
0.438	3FF80000
0.375	3FF00000
0.313	3FE80000
0.250	3FE00000
0.188	3FC00000
0.125	3FA00000
0.063	3F800000
0.000	0
-0.063	C0A00000
-0.125	C0C00000
-0.188	C0500000
-0.250	C0200000
-0.313	C0180000
-0.375	C0100000
-0.438	C0080000
-0.500	BFE00000
-0.563	BFEC0000
-0.625	BFE80000
-0.688	BFE40000
-0.750	BFE00000
-0.813	BFCC0000
-0.875	BFC80000
-0.938	BFC40000